A Survey of Evolutionary Continuous Dynamic Optimization Over Two Decades

Yazdani, Danial; Cheng, Ran; Yazdani, Donya; Branke, Jurgen; Jin, Yaochu; Yao, Xin

Published in:
IEEE Transactions on Evolutionary Computation

DOI:
10.1109/TEVC.2021.3060012

Publication date:
2021

Citation for published version (APA):
Yazdani, D., Cheng, R., Yazdani, D., Branke, J., Jin, Y., & Yao, X. (2021). A Survey of Evolutionary Continuous Dynamic Optimization Over Two Decades: Part B. *IEEE Transactions on Evolutionary Computation*, *25*(4), 630-650. [9356720]. https://doi.org/10.1109/TEVC.2021.3060012

A Survey of Evolutionary Continuous Dynamic Optimization Over Two Decades – Part B

Danial Yazdani, *Member, IEEE,* Ran Cheng, *Member, IEEE,* Donya Yazdani, Jürgen Branke, *Member, IEEE,* Yaochu Jin, *Fellow, IEEE,* and Xin Yao, *Fellow, IEEE,*

***Abstract*—This paper presents the second part of a two-part survey that reviews evolutionary dynamic optimization for single-objective unconstrained continuous problems over the last two decades. While in the first part we reviewed the components of dynamic optimization algorithms, in this part, we present an in-depth review of the most commonly used benchmark problems, performance analysis methods, static optimization methods used in the framework of dynamic optimization algorithms, and real-world applications. Compared to the previous works, this paper provides a new taxonomy for the benchmark problems used in the field based on their baseline functions and dynamics. In addition, this survey classifies the commonly used performance indicators into fitness/error based and efficiency based ones. Different types of plots used in the literature for analyzing the performance and behavior of algorithms are also reviewed. Furthermore, the static optimization algorithms which are modified and utilized in the framework of dynamic optimization algorithms as the optimization components are covered. We then comprehensively review some real-world dynamic problems which are optimized by evolutionary dynamic optimization methods. Finally, some challenges and opportunities are pointed out for future directions.**

***Index Terms*—Unconstrained continuous dynamic optimization, Evolutionary algorithms, Dynamic benchmark problems, Performance indicators, Continuous dynamic real-world problems, Future directions.**

P_B-I. Introduction

In the first part of this survey [1], we provided a detailed review of the components of dynamic optimization algorithms (DOAs) developed for single-objective unconstrained continuous dynamic optimization problems (DOPs). In Part B, we comprehensively review:

- DOP benchmarks,
- Performance analysis methods,
- Optimization components, and
- Real-world DOPs.

We then provide an in-depth discussion on the potential future directions in the field.

A DOP benchmark is usually proposed as a package of a *baseline function* that generates the landscape and some *dynamics* that change the search space, e.g., by altering the baseline function parameter values. Although dynamics and baseline functions are usually not dependent on each other, up to now, they have never been reviewed separately. Herein, we provide a detailed review of the baseline functions and dynamics of the DOP benchmarks separately. This potentially helps the readers to pick desired combinations of a baseline function and dynamics and to generate more diverse DOP problem instances. Furthermore, we propose to classify the baseline functions used in the DOP benchmarks into basic static functions, moving peaks based functions, and the composition of basic static functions. Among the above-mentioned baseline functions, *moving peaks based functions* are specifically designed for DOP benchmarks. The generated landscapes by moving peaks baseline functions are constructed by assembling several peaks. The number and also some other attributes of these peaks, such as their locations and heights, are controllable.

We then review the commonly used performance indicators and plots that have been used for analyzing and comparing the performance of DOAs. We classify DOP performance indicators into fitness/error based and efficiency based ones. The fitness/error based indicators measure the performance of DOAs according to the obtained solutions' fitness/error over specific points of time (e.g., at the end of each environment). On the other hand, the efficiency based indicators calculate the efficiency of DOAs based on some specific aspects, such as tracking efficiency according to the average distance to optimum [2], and peak coverage efficiency based on the number of discovered peaks [3].

Then, we review the static optimization algorithms (with an emphasis on the evolutionary and swarm intelligence algorithms) used as the optimization component in DOAs. This survey then provides a comprehensive review of real-world DOPs that have been solved by DOAs reviewed in the first part of this survey.

Finally, we point out some remaining issues and limitations of the field which cause a gap between academic research and real-world applications. We also indicate some important

This work was supported by the Shenzhen Peacock Plan (Grant No. KQTD2016112514355531), the Guangdong Provincial Key Laboratory (Grant No. 2020B121201001), the National Natural Science Foundation of China (Grant Nos. 61903178, 61906081, and U20A20306), the Program for Guangdong Introducing Innovative and Enterpreneurial Teams (Grant No. 2017ZT07X386), and the Program for University Key Laboratory of Guangdong Province (Grant No. 2017KSYS008). *(Corresponding author: Ran Cheng.)*

Danial Yazdani, R. Cheng and X. Yao are with Guangdong Provincial Key Laboratory of Brain-inspired Intelligent Computation, Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China (e-mails: danial.yazdani@gmail.com, chengr@sustc.edu.cn, xiny@sustc.edu.cn). X. Yao is also with the CERCIA, School of Computer Science, Birmingham B15 2TT, United Kingdom.

Donya Yazdani is with the Advanced Reasoning Group, Department of Computer Science, Aberystwyth University, Aberystwyth, Ceredigion SY23 3DB, United Kingdom (email: d.yazdani@aber.ac.uk).

J. Branke is with the Operational Research and Management Sciences Group in Warwick Business school, University of Warwick, Coventry CV4 7AL, United Kingdom (email: Juergen.Branke@wbs.ac.uk).

Y. Jin is with Department of Computer Science, University of Surrey, Guildford, Surrey GU27XH, United Kingdom (email: yaochu.jin@surrey.ac.uk).

topics in the area which have not received enough attention. Thereafter, we suggest several important potential future directions.

The organization of the rest of this paper is as follows. The commonly used benchmark problems in the DOP literature are reviewed in Section (§) $\mathrm{P_B}$-II[1]. The performance indicators and plots used for analyzing performance of DOAs are reviewed in § $\mathrm{P_B}$-III. Static optimization algorithms used as the optimization components in DOAs are reviewed in § $\mathrm{P_B}$-IV. § $\mathrm{P_B}$-V reviews real-world DOPs. § $\mathrm{P_B}$-VI provides some potential future research directions. Finally, § $\mathrm{P_B}$-VII concludes the second part of the survey paper.

$\mathrm{P_B}$-II. Benchmark problems

To assess the effectiveness of DOAs, they are usually tested on different generated problem instances by DOP benchmarks. Selection of proper benchmark generators with user-defined dynamics and landscape characteristics is essential for examination of DOAs. In this survey, we review the DOP benchmarks regarding their baseline functions which generate landscapes, and their dynamics which control environmental changes.

A. Baseline functions of DOP benchmarks

The baseline functions of the DOP benchmarks can be classified into three groups:

- Basic static functions, such as Sphere, Ackley, Rastrigin, Rosenbrock, and Griewank,
- Composition of basic static functions [4], [5], which are used in the generalized dynamic benchmark generator (GDBG) [6], and
- Moving peaks baseline functions which can generate landscapes with a controllable number of peaks (components). The attributes of each peak, such as location and height, are configurable in these functions.

Table $\mathrm{P_B}$-I shows where these baseline functions are used for DOAs performance assessment. As can be seen in this table, moving peaks baseline functions are the most popular ones and are used in more than 70% of the DOP literature. The reason is that they are easy to understand/implement and highly configurable/controllable.

In the rest of this subsection, we review the moving peaks baseline functions. Note that the baseline functions of the basic static functions and their compositions are out of the scope of this survey since they are originally designed for static optimization.

1) Moving peaks baseline functions: Generated landscapes by moving peaks baseline functions are constructed by assembling several components. A $\max(\cdot)$ function is normally used to define the basin of attraction of components. Every component in moving peaks baseline functions usually contains a *peak* whose attributes such as height, and location change over time. The moving peaks benchmark (MPB) [28] and DF1 [124] are the first DOP benchmarks that use moving peaks baseline functions. The baseline function of MPB is defined as below:

$$f^{(t)}(\vec{x}) = \max_{i\in\{1,\ldots,m\}} \frac{h_i^{(t)}}{1 + w_i^{(t)} \sum_{j=1}^{d} \left(x_j - c_{i,j}^{(t)}\right)^2}, \quad (\mathrm{P_B}\text{-}1)$$

where m is the number of peaks, x_j is the jth dimension of a solution ($\vec{x}$) in a d-dimensional problem space, $c_{i,j}^{(t)}$ is the jth dimension of the center of ith peak in the tth environment ($\vec{c}_i^{\,(t)}$) , and $h_i^{(t)}$ and $w_i^{(t)}$ are the height and width of the ith peak in the tth environment, respectively.

DF1 uses a baseline function that generates landscapes with conical peaks:

$$f^{(t)}(\vec{x}) = \max_{i\in\{1,\ldots,m\}} \left\{ h_i^{(t)} - w_i^{(t)} \left\| \vec{x} - \vec{c}_i^{\,(t)} \right\| \right\}. \quad (\mathrm{P_B}\text{-}2)$$

($\mathrm{P_B}$-2) is also used in the second version of MPB, which is known as *Scenario 2* in the literature [178].

In [133], ($\mathrm{P_B}$-2) is modified by adding a threshold β to determine the minimum fitness value of the problem:

$$f^{(t)}(\vec{x}) = \max\left\{ \beta, \max_{i\in\{1,\ldots,m\}} \left\{ h_i^{(t)} - w_i^{(t)} \left\| \vec{x} - \vec{c}_i^{\,(t)} \right\| \right\} \right\}. \quad (\mathrm{P_B}\text{-}3)$$

In this baseline function, β is used to create plateau regions whose fitness values are equal to β.

Gaussian peaks benchmark (GPB) [132] uses another moving peaks baseline function that is capable of generating Gaussian peaks:

$$f^{(t)}(\vec{x}) = \max_{i\in\{1,\ldots,m\}} \left\{ h_i^{(t)} \exp\left(-\frac{\left\| \vec{x} - \vec{c}_i^{\,(t)} \right\|^2}{2\left(w_i^{(t)}\right)^2} \right) \right\}. \quad (\mathrm{P_B}\text{-}4)$$

In [6], a dynamic rotation peak benchmark generator (DRPBG) is proposed whose baseline function is:

$$f^{(t)}(\vec{x}) = \max_{i\in\{1,\ldots,m\}} \frac{h_i^{(t)}}{1 + w_i^{(t)} \sqrt{\sum_{j=1}^{d} \frac{(\vec{x}_j - \vec{c}_{i,j}^{\,(t)})^2}{d}}}. \quad (\mathrm{P_B}\text{-}5)$$

This baseline function is used as one of the scenarios (F_1) of GDBG and has been frequently used in the DOP literature.

TABLE $\mathrm{P_B}$-I
BASELINE FUNCTIONS IN DOP LITERATURE.

Baseline function	References
Basic static baseline functions[a]	[2], [7]–[27]
Moving peaks baselines[b]	[3], [15], [18], [23], [25]–[173]
Composition of basic static functions[c]	[6], [75], [97], [106], [111], [114], [115], [146]–[148], [150]–[172], [174]
Others[d]	[24], [175]–[177]

[a] Basic static functions include Sphere, Ackley, Rastrigin, Rosenbrock, and Griewank.
[b] Including all baseline functions that generate a controllable number of peaks whose locations can change over time.
[c] Dynamic composition benchmark generator (CDBG) [6].
[d] Generated landscapes in these benchmarks have a controllable number of peaks (or valleys); however, their positions cannot change.

[1]Labels of all sections, equations, tables, and figures in Part A and Part B of this survey are prefixed with $\mathrm{P_A}$ and $\mathrm{P_B}$, respectively.

Generalized moving peaks benchmark (GMPB) [173] is another benchmark generator whose baseline function generates landscapes that are constructed by assembling several components. Unlike other moving peaks baseline functions, generated components by GMPB's baseline function can vary from smooth to highly irregular, unimodal to multimodal, and symmetric to asymmetric. They also can have circular contour lines or can be highly ill-conditioned. GMPB uses the following baseline function:

$$f^{(t)}(\vec{x}) = \max_{i\in\{1,\ldots,m\}} \left\{ h_i^{(t)} - \sqrt{\mathbb{T}\left(\left(\vec{x}-\vec{c}_i^{\,(t)}\right)^\top \mathbf{R}_i^{(t)\top}, i\right) \mathbf{W}_i^{(t)} \mathbb{T}\left(\mathbf{R}_i^{(t)}\left(\vec{x}-\vec{c}_i^{\,(t)}\right), i\right)} \right\}, \tag{$\mathrm{P_B}$-6}$$

where $\mathbb{T}(\mathbf{y}, i) : \mathbb{R}^d \mapsto \mathbb{R}^d$ is calculated as:

$$\mathbb{T}(y_j, i) = \begin{cases} \exp\left(\log(y_j) + \tau_i^{(t)}\left(\sin\left(\eta_{i,1}^{(t)}\log(y_j)\right) + \sin\left(\eta_{i,2}^{(t)}\log(y_j)\right)\right)\right) & \text{if } y_j > 0 \\ 0 & \text{if } y_j = 0 \\ -\exp\left(\log(|y_j|) + \tau_i^{(t)}\left(\sin\left(\eta_{i,3}^{(t)}\log(|y_j|)\right) + \sin\left(\eta_{i,4}^{(t)}\log(|y_j|)\right)\right)\right) & \text{if } y_j < 0 \end{cases} \tag{$\mathrm{P_B}$-7}$$

where $\mathbf{R}_i^{(t)}$ is the rotation matrix of ith component in tth environment, $\mathbf{W}_i^{(t)}$ is a $d \times d$ diagonal matrix whose diagonal elements show the width of ith component in different dimensions, y_j is jth dimension of $\mathbf{y}$, and $\eta_{i,k\in\{1,2,3,4\}}^{(t)}$ and $\tau_i^{(t)}$ are irregularity parameters of the ith component. Similar to the other moving peaks baseline functions, location, height, and width of each component are controllable in GMPB. In addition, components of GMPB can have different width values in different dimensions; therefore, the condition number[2] of components is controllable. Using a rotation matrix for each component, the degree of variable interactions in each component is adjustable. Also, the irregularity degree and modality can be controlled using $\eta_{k\in\{1,2,3,4\}}^{(t)}$ and $\tau^{(t)}$. Finally, by setting different values for $\eta_{k\in\{1,2,3,4\}}^{(t)}$ the symmetry of the components can be decided (identical values for all four η_k generate symmetric components).

Li et al. propose Free Peaks benchmark (FPs) [179], which generates landscapes containing multiple moving peaks. FPs divides the landscape by hypercubes, which determine the basin of attractions of peaks. This property of FPs is different from other moving peaks baseline functions that use $\max(\cdot)$ function to determine the basin of attractions. In each hypercube, there exists one peak whose boundaries are limited by this hypercube. The position of a component in a hypercube, and also hypercube's boundaries are controllable. Several single peak baseline functions, such as the conical peak, are suggested in [179] to be used in each hypercube.

All moving peaks baseline functions are scalable. On the one hand, if the number of peaks is set to one, the generated landscape by moving peaks baseline functions is fully separable [145]. On the other hand, landscapes with more than one peak are fully non-separable [180]. Consequently, they are not capable of producing modular (e.g., partially separable) problem instances, which makes them unsuited for generating large-scale problems [181]. To address this limitation, some works use *composition* approaches to generate modular problems. High-dimensional MPB (HDMPB) [144], composition MPB (CMPB) [145], and GMPB [173] generate modular problem instances by summing multiple subfunctions. In HDMPB and CMPB, the fitness function is constructed by summing several ($\mathrm{P_B}$-2) as subfunctions, while GMPB uses ($\mathrm{P_B}$-6) to do the same. In CMPB and GMPB, the parameters of subfunctions, such as the number of peaks and dimensions, can be different, which results in generating heterogeneous modular problem instances. In addition, in CMPB and GMPB, each subfunction is multiplied to a weight parameter to generate imbalance patterns. A consequence of composing several moving peaks baseline functions is an exponential growth in the number of peaks [145], [173], [180]. The number of peaks in the landscape of a composition function can be as large as the product of the numbers of peaks in all subfunctions.

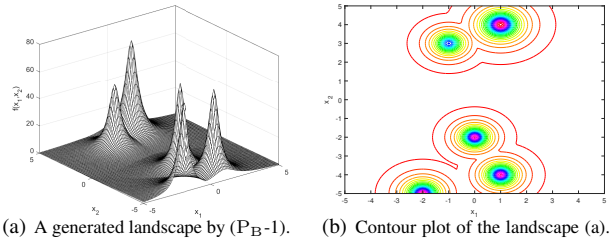
To highlight the differences of landscapes generated by different moving peaks baseline functions, we show some example landscapes created by each baseline function in Figures $\mathrm{P_B}$-1 and $\mathrm{P_B}$-2. Figure $\mathrm{P_B}$-1 shows generated landscapes by ($\mathrm{P_B}$-1), ($\mathrm{P_B}$-2), ($\mathrm{P_B}$-3), ($\mathrm{P_B}$-4), and ($\mathrm{P_B}$-5) with five peaks (i.e., components) and identical parameter values for height, and width values. The search range (i.e., variable domains) for ($\mathrm{P_B}$-2), ($\mathrm{P_B}$-3), and ($\mathrm{P_B}$-4) is set to $[-50, 50]$, and is set to $[-5, 5]$ for ($\mathrm{P_B}$-1) and ($\mathrm{P_B}$-5). We used the same random seed numbers to generate peak positions in all baseline functions according to their search ranges. By looking at Figure $\mathrm{P_B}$-1 and considering the baseline formulas, we make the following observations:

- In all problems, the optimum fitness value is equal to the largest height, and the optimum is positioned in the center of the peak with the largest height.
- Fitness values of all positions in ($\mathrm{P_B}$-1), ($\mathrm{P_B}$-4), and ($\mathrm{P_B}$-5) are greater than zero. ($\mathrm{P_B}$-2) generates landscapes that usually contain areas with negative fitness values. Also, the minimum fitness value in the generated landscapes by ($\mathrm{P_B}$-3) is equal to β.
- The gradient in the generated flat looking areas by ($\mathrm{P_B}$-1), ($\mathrm{P_B}$-4), and ($\mathrm{P_B}$-5), is non-zero. Therefore, DOAs do not face the challenges posed by plateau areas in the aforementioned landscapes. However, ($\mathrm{P_B}$-3) generates plateau areas which are very challenging for DOAs. Such plateaus can trap individuals.
- Peaks generated by ($\mathrm{P_B}$-1) and ($\mathrm{P_B}$-5) are much narrower than those generated by others using the same width values. Although we have shrunk the search range to $[-5, 5]$ in the generated landscapes by these two baseline functions, their peaks still remain much narrower than those of others.
- Exploitation and tracking in the landscapes generated by ($\mathrm{P_B}$-1) and ($\mathrm{P_B}$-5) (with the same width values) are relatively more challenging for DOAs as the areas around the local optima are very sharp and even close positions to the optimum can have large differences in fitness values.

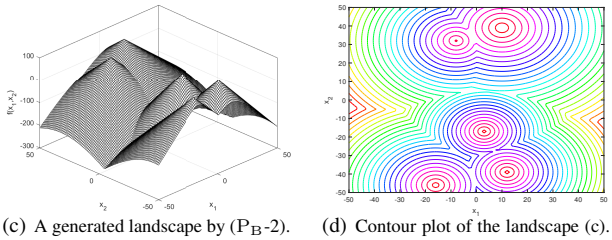
Figure $\mathrm{P_B}$-2 depicts some sample landscapes that are gen-

[2]The ratio of the largest width value of a component to its smallest value indicates its condition number [173]. If a component's width value is stretched in one axis's direction more than the other axes, then, the component is ill-conditioned.

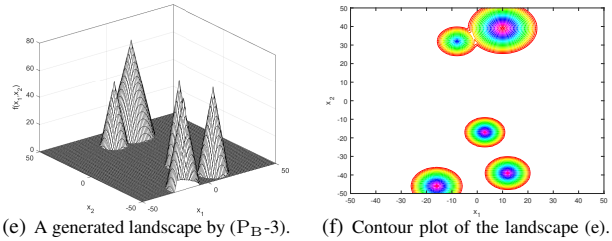
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2021.3060012, IEEE Transactions on Evolutionary Computation



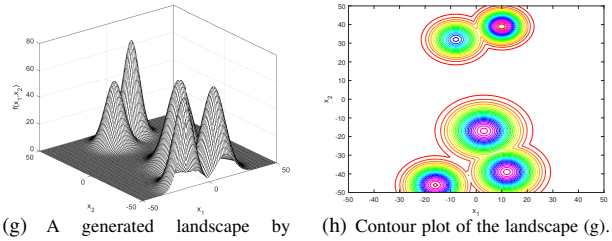
(a) A generated landscape by ($\mathrm{P_B}$-1). (b) Contour plot of the landscape (a).



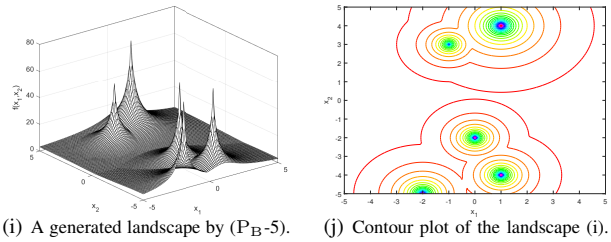
(c) A generated landscape by ($\mathrm{P_B}$-2). (d) Contour plot of the landscape (c).



(e) A generated landscape by ($\mathrm{P_B}$-3). (f) Contour plot of the landscape (e).



(g) A generated landscape by ($\mathrm{P_B}$-4). (h) Contour plot of the landscape (g).

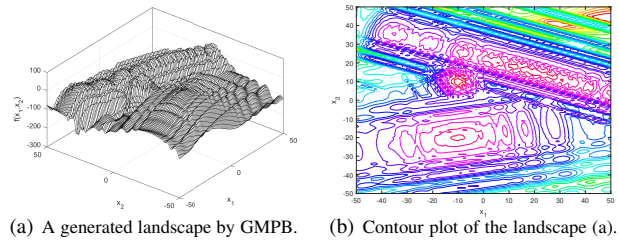


(i) A generated landscape by ($\mathrm{P_B}$-5). (j) Contour plot of the landscape (i).

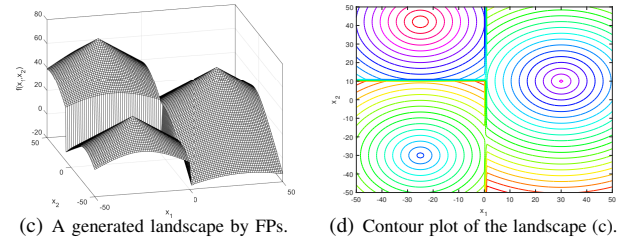
Fig. $\mathrm{P_B}$-1. Generated landscapes with five peaks by the baseline functions ($\mathrm{P_B}$-1), ($\mathrm{P_B}$-2), ($\mathrm{P_B}$-3) with $\beta = 0$, ($\mathrm{P_B}$-4), and ($\mathrm{P_B}$-5). Each landscape's contour plot in the left column is shown in the same row of the right column. The height (uniformly randomized in $[30, 70]$), width (uniformly randomized in $[1, 12]$), and location of all peak center positions (uniformly randomized with the same random seed numbers in $[-50, 50]$ for ($\mathrm{P_B}$-2), ($\mathrm{P_B}$-3), ($\mathrm{P_B}$-4) and in $[-5, 5]$ for ($\mathrm{P_B}$-1) and ($\mathrm{P_B}$-5)) are similar in all landscapes.

erated by GMPB, FPs, and CMPB/HDMPB. Figure $\mathrm{P_B}$-2(a) shows a generated landscape by GMPB's baseline function ($\mathrm{P_B}$-6), which is constructed by assembling *three* components[3]. Unlike all illustrated landscapes in Figure $\mathrm{P_B}$-1, this landscape is highly multimodal. The global optimum is positioned at the center of the component with the largest height

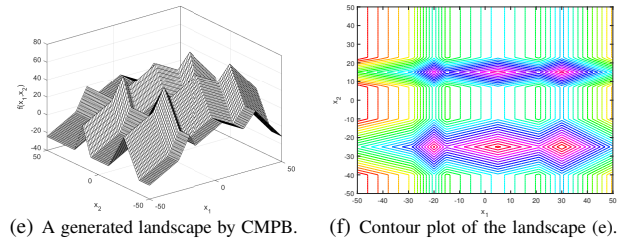
[3]For GMPB, we cannot call them peak as they are highly multimodal and contain many smaller peaks in their basin of attraction.



(a) A generated landscape by GMPB. (b) Contour plot of the landscape (a).



(c) A generated landscape by FPs. (d) Contour plot of the landscape (c).



(e) A generated landscape by CMPB. (f) Contour plot of the landscape (e).

Fig. $\mathrm{P_B}$-2. Generated landscapes by GMPB [173], FPs [179], and CMPB/HDMPB [144], [145]. The contour plot of each landscape in the left column is shown in the right column. In (a), a generated landscape by GMPB ($\mathrm{P_B}$-6) with three highly irregular, rotated, ill-conditioned (two of components), and asymmetric components is shown. (c) shows a generated landscape by FPs with three *conical* peaks whose basin of attractions are determined by their hypercubes. Finally, (e) illustrates a generated landscape by CMPB/HDMPB, which is constructed by composing two 1-dimensional landscapes with two and three peaks which are generated by ($\mathrm{P_B}$-2).

value. All three generated components by GMPB are irregular and asymmetric. Moreover, two of the components are ill-conditioned and rotated. Note that in GMPB, we can control the intensity of different characteristics of the components, and in the simplest form, the components will be conical similar to the components created by ($\mathrm{P_B}$-2).

Figure $\mathrm{P_B}$-2(c) illustrates how the landscapes divided by hypercubes in FPs are different from those of other moving peaks baseline functions with $\max(\cdot)$ function. As can be seen, there are sharp drops alongside the hypercube boundaries. Finally, Figure $\mathrm{P_B}$-2(e) shows a landscape generated by CMPB/HDMPB which is constructed by composing two 1-dimensional subfunctions with two and three conical peaks.

B. Dynamics in DOP benchmarks

In this subsection, we review the frequently used dynamics in DOP benchmarks. Each dynamic generates environmental changes with different characteristics. They control the severity of environmental changes (from gradual to abrupt) and their patterns (cyclic, reappearing, random, chaotic, and linear).

One of the first introduced dynamics is switching between landscapes [175]. By switching between a predefined set of landscapes, the landscapes will *reappear* over time. Pendulum MPB (PMPB) [138] is a DOP benchmark which uses this

feature. PMPB works based on a pendulum length parameter pl and a direction parameter dir. It first generates pl environments $\left[f^{(1)}, f^{(2)}, \cdots, f^{(pl)}\right]$. Then, it archives the parameters of these environments for future reappearances. When the moments of changes come, an environment is retrieved from the archive according to the pendulum movement over the list. For example, given that three environments $\{A, B, C\}$ are listed in the archive, their reappearance order will be $\{A, B, C, B, A, B, C, B, \cdots\}$.

Another dynamic used in the moving peaks based benchmarks is *peak shape change*. This dynamic affects each peak's shape and can change the fitness values of all solutions on the peak's basin of attraction except the peak summit. FPs [179] uses this dynamic to change the shape of peaks in hypercubes. In this benchmark, a predefined set of n basic functions $F = \{f_1, f_2, \cdots, f_n\}$ are defined, which can be used for generating each peak in each environment. In each environmental change, a basic function from F is randomly chosen to define each peak's shape in each hypercube. Note that this dynamic only switches the basic function of a hypercube and it does not affect other parameters such as peak center positions or hypercube boundaries. Similarly, this dynamic can be used for other moving peaks baseline functions, which results in a landscape with heterogeneous peaks in terms of shape.

Another commonly used dynamic is the landscape shifting. Angeline first introduced the idea of shifting the landscape of a static function to generate environmental changes in [16]. In this work, three types of dynamics, called linear, random, and circular, are applied to a 3-dimensional sphere function to create environmental changes by shifting the landscape. Given a severity parameter $\tilde{\phi}$, these dynamics are formulated as below:

$$\phi_i^{(t+1)} = \phi_i^{(t)} + \tilde{\phi}, \tag{$\mathrm{P_B}$-8}$$

$$\phi_i^{(t+1)} = \phi_i^{(t)} + \tilde{\phi}\mathcal{N}(0,1), \tag{$\mathrm{P_B}$-9}$$

$$\phi_i^{(t+1)} = \begin{cases} \phi_i^{(t)} + \tilde{\phi}\sin\dfrac{2\pi t}{25} & i = 1 \text{ and } 3, \\ \phi_i^{(t)} + \tilde{\phi}\cos\dfrac{2\pi t}{25} & i = 2, \end{cases} \tag{$\mathrm{P_B}$-10}$$

where $\phi_i^{(t)}$ is the offset of the ith dimension in tth environment for $i \in \{1, 2, 3\}$, $\tilde{\phi}$ is the shift severity, and $\mathcal{N}(0,1)$ is a random number drawn from a normal distribution with mean 0 and variance 1. By setting $\tilde{\phi}$ from small to large values, dynamics generate gradual to abrupt environmental changes. Although these dynamics have been used to shift the landscape in [16], they can be used for changing other environment's control parameters such as the width and height of peaks. Note that linear and circular dynamics generate *predictable patterns* [182]. The idea of shifting the search space is also expanded to the other static benchmark functions with higher dimensions [20]. As shown in Table $\mathrm{P_B}$-I, DOP benchmarks which shift the generated landscapes by basic static functions have been widely used in the literature.

DF1 [124] and MPB [28] are among the first proposed moving peaks baseline functions. The width, height, and center of each peak in these two benchmarks can change using some dynamics. DF1 uses a logistic function for changing the parameters of peaks. In MPB, inspired from ($\mathrm{P_B}$-9), the height and width of all peaks change from one environment to the next using the following equations:

$$h_i^{(t+1)} = h_i^{(t)} + \tilde{h}\mathcal{N}(0,1), \tag{$\mathrm{P_B}$-11}$$

$$w_i^{(t+1)} = w_i^{(t)} + \tilde{w}\mathcal{N}(0,1), \tag{$\mathrm{P_B}$-12}$$

where $\tilde{h}$ and $\tilde{w}$ are the severity values of height and width, respectively. Moreover, a dynamic is presented for relocating the peaks' center positions in MPB as follows:

$$\vec{c}_i^{\,(t+1)} = \vec{c}_i^{\,(t)} + \vec{v}_i^{\,(t+1)}, \tag{$\mathrm{P_B}$-13}$$

$$\vec{v}_i^{\,(t+1)} = \tilde{s} \cdot \frac{(1-\lambda)\cdot\vec{u} + \lambda\cdot\vec{v}_i^{\,(t)}}{\left\|(1-\lambda)\cdot\vec{u} + \lambda\cdot\vec{v}_i^{\,(t)}\right\|}, \tag{$\mathrm{P_B}$-14}$$

where $\tilde{s}$ is the severity shift, $\vec{u}$ is a vector of uniformly distributed numbers in $[-0.5, 0.5]$, and $\lambda \in (0,1)$ is the correlation coefficient. A choice of $\lambda = 1$ implies that the relocation of each peak's center position is fully correlated with its previous relocation (i.e., it moves in the same direction) whereas it is uncorrelated if $\lambda = 0$ (i.e., it moves toward a random direction in each environmental change). For ($\mathrm{P_B}$-14), $\|\vec{v}_i^{\,(t+1)}\|$ is always equal to $\tilde{s}$ [28]. Severity of environmental changes can be configured by setting $\tilde{h}$, $\tilde{w}$, and $\tilde{s}$. The larger the values of these parameters, the more severe environmental changes will be.

Six different types of dynamics are used in GDBG [183]: *small step, large step, random, chaotic, recurrent*, and *recurrent with noise*. These dynamics are formulated respectively as follows:

$$\Delta\phi = \gamma r\tilde{\phi}\|\phi\|, \tag{$\mathrm{P_B}$-15}$$

$$\Delta\phi = \tilde{\phi}\|\phi\|(\gamma\mathrm{sign}(r) + r(\gamma_{\max} - \gamma)), \tag{$\mathrm{P_B}$-16}$$

$$\Delta\phi = \tilde{\phi}\mathcal{N}(0,1), \tag{$\mathrm{P_B}$-17}$$

$$\phi^{(t+1)} = A\phi^{(t)}\frac{1-\phi^{(t)}}{\|\phi\|}, \tag{$\mathrm{P_B}$-18}$$

$$\phi^{(t+1)} = \phi_{\min} + \frac{\|\phi\|}{2}(\sin(\frac{2\pi}{p}t + \varphi) + 1), \tag{$\mathrm{P_B}$-19}$$

$$\phi^{(t+1)} = \phi_{\min} + \frac{\|\phi\|}{2}(\sin(\frac{2\pi}{p}t + \varphi) + 1) + \tilde{n}\mathcal{N}(0,1), \tag{$\mathrm{P_B}$-20}$$

where $\Delta\phi$ is the deviation from the current control parameter values, $\|\phi\|$ is the change range of ϕ, $\phi^{(t)}$ is the offset in tth environment, $\tilde{\phi} \in (0,1)$ is the change severity of ϕ, $\phi_{\min}$ is the minimum value of ϕ, $\tilde{n} \in (0,1)$ is the noise severity, $\gamma, \gamma_{\max} \in (0,1)$ and A are constant values, $r \in (-1,1)$ is a random number drawn from a uniform distribution, p is the period number, and φ is the initial phase. It is worth mentioning that the *random* dynamic in ($\mathrm{P_B}$-17) is similar to the ones in ($\mathrm{P_B}$-9), ($\mathrm{P_B}$-11), and ($\mathrm{P_B}$-12). Moreover, the logistic function used for *chaotic* dynamic in ($\mathrm{P_B}$-18) is similar to the one used in DF1. GDBG [183] uses the dynamics defined in ($\mathrm{P_B}$-15) to ($\mathrm{P_B}$-20) for two groups of baseline functions: DRPBG and CDBG. In CDBG, the position and fitness offsets of each subfunction, which is a basic static function, are changed using GDBG's dynamics. In DRPBG,

GDBG's dynamics are used to change the width and height of peaks. Furthermore, the peaks' center positions in DRPBG change using rotation matrices introduced in [184].

In [183], a dynamic is proposed to change the problem dimensionality. This dynamic can also be used in other DOP benchmarks to change their number of dimensions (i.e., number of variables). However, it should be mentioned that in most existing DOP benchmarks, by adding or removing a dimension, the optimum position in the old dimensions (assuming there is no other type of change) remains unchanged. Consequently, using this dynamic for reducing the number of dimensions does not pose any specific challenge in most existing DOP benchmarks. On the other hand, by increasing the number of dimensions, the DOAs only need to optimize the newly added dimensions (assuming that the previous global optimum was discovered).

In [133], a control parameter is used to determine the percentage of peaks to change at each environmental change (peaks are chosen randomly with uniform distribution). This process makes the change detection more challenging for some change detection components (see §$\mathrm{P_A}$-III-B) as only some random parts of the environment change [134].

A dynamic is introduced in [142] that changes the number of peaks over time. Appearance of a new peak in the landscape can be challenging for change detection components (see §$\mathrm{P_A}$-III-B) since it usually causes a locally environmental change. This can be even more challenging if this new peak contains the global optimum.

A general framework for inducing artificial changes (IAC) in optimization problems is proposed in [185]. IAC uses different dynamics including 1) shift, 2) rotation, 3) decision variables permutation, and 4) duplication based dynamics where a predefined number of decision variables are randomly copied to other positions. These dynamics can be applied to any landscape to generate local and global environmental changes.

TABLE $\mathrm{P_B}$-II
DOP BENCHMARKS THAT USE DIFFERENT MOVING PEAKS BASELINE FUNCTIONS WITH DIFFERENT TYPES OF DYNAMICS, AND THEIR CORRESPONDING REFERENCES.

Version	References
Moving peaks benchmark (MPB)[a]	[15], [18], [23], [25]–[28], [30]–[64], [67]–[123], [155], [156], [166]
DF1[b]	[3], [81], [124]–[130]
Moving valleys benchmark (MVB)[c]	[131]
Gaussian peaks benchmark (GPB)	[15], [132]
MPBs with local environmental changes[d]	[133], [134]
MPBs with cyclic and pendulum changes[e]	[135]–[138]
Multimodal MPB[f]	[139]
MPBs with varying number of peaks	[140]–[143]
MPBs whose peaks have different change severity[g]	[186]
Constrained MPBs[h]	[187], [188]
Modular MPBs[i]	[144], [145], [173]
DRPBG[j]	[6], [75], [97], [106], [111], [114], [115], [146]–[148], [150]–[172], [174]
Free peaks[k]	[179]
Generalized MPB[l]	[173]

[a] Including Scenarios 1 and 2 of MPB.
[b] DF1 uses the same baseline function as MPB's Scenario 2, but it uses chaotic dynamics.
[c] A minimization version of MPB with spherical valleys.
[d] A portion of peaks are involved in the environmental changes, which results in generating local changes, which are more challenging to detect.
[e] The optima reapper in their previous locations, making these benchmark problems suitable for testing explicit memory/prediction-based methods [28].
[f] Suitable for multimodal optimization where there exist multiple global optima.
[g] Suitable for robust optimization over time (ROOT) [189]. Each peak has its own shift, height, and width severity values which result in generating peaks with different levels of robustness.
[h] Using two moving peaks baseline functions (one as the objective function and the other one as constraint), these benchmark suites generate constrained problems with multiple moving disjointed feasible regions.
[i] These problems are suitable to generate large-scale DOPs.
[j] Using GDBG's [6] dynamics to change height and width of peaks, and rotation transforms [184] to change peaks' locations.
[k] In Free peaks, the landscape is divided by hypercubes which each contains a peak.
[l] Generalized MPB is capable of generating modular problem instances whose components have a variety of features including different levels of variable interaction, ill-conditioning, irregularity, and symmetry.

C. Discussion on DOP benchmarks

In this section, we provided a review of the DOP benchmarks in the field based on their baseline functions and dynamics. As shown in Table $\mathrm{P_B}$-I, moving peaks baseline functions are the most commonly used benchmarks in the DOP literature. We have comprehensively reviewed moving peaks baseline functions in § $\mathrm{P_B}$-II-A1. Although the corresponding references of moving peaks baseline functions use a limited number of dynamics, a variety of dynamics that were reviewed in § $\mathrm{P_B}$-II-B can be applied to them to provide more diverse types of environmental changes. Table $\mathrm{P_B}$-II lists different DOP benchmarks that use moving peaks baseline functions with different types of dynamics. This table makes it easier for readers to find the works that use DOP benchmarks (moving peaks) with some specific desired characteristics.

Among all DOP benchmarks, the state-of-the-art GMPB [173] is the most configurable benchmark problem. GMPB generates modular problem instances, i.e., the instances which are fully-separable, partially-separable, and not-separable, where the former two can exhibit heterogeneity and imbalance properties. Such features make GMPB suitable for generating low-dimensional to large-scale problem instances. In addition, generated components by GMPB can range from smooth, unimodal, symmetric, separable, and with circular contour lines to highly irregular, multimodal, asymmetric, fully non-separable, and ill-conditioned. Thus, GMPB poses new problem characteristics that have not been captured in other DOP benchmarks. The empirical studies in [173] indicate that the problem instances generated by GMPB are quite challenging for the existing DOAs.

In addition to the baseline function and dynamic(s) of a DOP, change severity (i.e., spatial severity) and change frequency (i.e., temporal severity) are other two important factors that determine the characteristics and difficulty of a DOP. As can be seen in the most of the dynamics in § $\mathrm{P_B}$-II-B, there is a *severity parameter* that controls the change severity. Some examples of such parameters are $\tilde{\phi}$ in some Angelin's and GDBG's dynamics and $\tilde{s}$ in MPB. The larger the values

of these severity parameters, the more severe environmental changes will be. These parameters can also be set to have different values for different peaks to generate heterogeneous environmental change severity across the landscape [186]. On the other hand, the change frequency determines when the environmental changes should happen by applying dynamics to the environmental parameters. In DOP benchmarks, the change frequency parameter ϑ determines the number of fitness evaluations in each environment, i.e., when a DOA evaluates ϑ solutions, an environmental change will happen. Consequently, smaller values of ϑ result in greater change frequency/temporal severity, and vice versa. DOPs with higher change frequencies (i.e., smaller values of ϑ) are challenging due to the limited available computational resources between environmental changes.

By specifying different values for the parameters relevant to change severity and frequency, we can generate problem instances with different characteristics (see § $\mathrm{P_A}$-II), such as progressive, abrupt, and chaotic DOPs [95].

$\mathrm{P_B}$-III. Performance analysis methods

Measuring the performance of DOAs is crucial in their development. By using performance analysis methods, researchers are able to assess the effectiveness of DOAs, analyze their behavior in different situations, and compare their performance to the existing methods from different aspects. In the DOP literature, the performance of a DOA is usually analyzed using *plots* and *performance indicators*. The most commonly used plots demonstrate the convergence behavior of DOAs and the quality of the obtained solutions over time in terms of fitness or error. The performance indicators, on the other hand, show the efficiency of DOAs according to some specific criteria such as average error/fitness of the best found solutions over some predefined points of time. In the following of this section, we provide detailed descriptions of the well-known performance indicators and plots in the field.

A. Performance indicators

We classify the DOP performance indicators into two classes: fitness/error based and efficiency based performance indicators. The fitness/error based group includes indicators that measure the performance of DOAs according to the fitness/error of the obtained solutions. The efficiency based indicators, on the other hand, calculate the efficiency of DOAs based on some specific aspects, e.g., tracking efficiency according to the average distance to optimum [2], and peak coverage efficiency according to the number of discovered peaks [3].

1) Fitness/error based performance indicators: If the information of the global optimum in each environment is available, then the error of solutions can be calculated. Although in the real-world applications the information of the global optimum is not available, this information is accessible in most DOP benchmarks.

One performance indicator that needs the information of the global optimum is *offline-error* [178] (E_O). It is the most commonly used performance indicator in the literature. This approach calculates the average error of the best found position over all fitness evaluations using the following equation:

$$E_O = \frac{1}{T\vartheta}\sum_{t=1}^{T}\sum_{c=1}^{\vartheta}\left(f^{(t)}\left(\vec{x}^{\star(t)}\right) - f^{(t)}\left(\vec{x}^{*((t-1)\vartheta+c)}\right)\right), \tag{$\mathrm{P_B}$-21}$$

where $\vec{x}^{\star(t)}$ is the global optimum position at the tth environment, T is the number of environments, ϑ is the change frequency, c is the fitness evaluation counter for each environment, and $\vec{x}^{*((t-1)\vartheta+c)}$ is the best found position at the cth fitness evaluation in the tth environment. In some works, a different version of the offline-error is used, where the errors over all *iterations* are considered instead of errors over all fitness evaluations. Since this version of the offline-error works based on iterations, it is not usually suitable for comparing the performance of DOAs whose optimization components, resource allocation methods, and parameter settings are different. In such circumstances, the number of iterations that algorithms run in each environment can be considerably different. Consequently, using an iteration based offline-error, we cannot have a fair comparison in many cases.

E_{BBC} [29] is another fitness/error based performance indicator that only considers the last error before each environmental change (i.e., at the end of each environment):

$$E_{\mathrm{BBC}} = \frac{1}{T}\sum_{t=1}^{T}\left(f^{(t)}\left(\vec{x}^{\star(t)}\right) - f^{(t)}\left(\vec{x}^{*(t)}\right)\right), \tag{$\mathrm{P_B}$-22}$$

where $\vec{x}^{*(t)}$ is the best found position in tth environment which is fetched at the end of the environment. However, using the quality of solutions at the end of environments is not helpful in realistic circumstances. Indeed, what matters is the quality of solutions in each environment at the *deployment time*, which is when a new solution is fetched to be deployed/implemented. In many real-world DOPs, DOAs need to find a better solution after each environmental change according to a deadline that is defined based on the system tolerance, which is denoted as *quick recovery* [190]. To address the aforementioned shortcoming of E_{BBC}, the point of fetching the best found position is suggested to be the time of deployment of a new solution instead of the end of environments [180]:

$$E_{\mathrm{BBD}} = \frac{1}{T}\sum_{t=1}^{T}\left(f^{(t)}\left(\vec{x}^{\star(t)}\right) - f^{(t)}\left(\vec{x}^{*(t-1)*\vartheta+\eta}\right)\right), \tag{$\mathrm{P_B}$-23}$$

where η shows the deployment time of the new solution after each environmental change, and $\vec{x}^{*(t-1)*\vartheta+\eta}$ is the the best found position after η fitness evaluations since the beginning of the tth environment.

Up until now, all the described performance indicators needed information about the position and fitness of the global optimum which is unsuitable for real-world DOPs. Offline-performance (P_O) [28], on the contrary, does not need such information. This indicator uses the average fitness of the best

found position over all fitness evaluations which is calculated by:

$$P_O = \frac{1}{T\vartheta}\sum_{t=1}^{T}\sum_{c=1}^{\vartheta} f^{(t)}\left(\vec{x}^{*((t-1)\vartheta+c)}\right). \qquad (\mathrm{P_B}\text{-}24)$$

Unlike the error based performance indicators whose ideal values are zero (i.e., no error), P_O is dependent on the fitness values of the global optimum in different environments which makes it unsuitable for fair comparisons between different algorithms. For example, an algorithm with zero error in an environment whose optimum value is 40, is considered worse than an algorithm with an error of 5 in an environment whose optimum value is 50. To have a fair comparison among algorithms using this performance indicator, the environmental parameters of the problem (e.g., heights, widths, and locations of peaks in moving peaks baseline functions) in all environments must be identical for all algorithms.

Other fitness/error based performance indicators include collective mean fitness [191], average local errors [128], and accuracy [192]. While the *collective mean fitness* calculates the average of the best found solution's fitness value over all iterations in all runs, the average local errors calculates an average error according to the peak coverage by DOAs for all peaks. Accuracy performance indicator, shown by E_A is calculated by:

$$E_A = \frac{1}{T\vartheta}\sum_{t=1}^{T}\sum_{c=1}^{\vartheta}\frac{f^{(t)}\left(\vec{x}^{*(t-1)*\vartheta+c}\right) - f_{\min}^{(t)}}{f_{\max}^{(t)} - f_{\min}^{(t)}}, \qquad (\mathrm{P_B}\text{-}25)$$

where $f_{\min}^{(t)}$ and $f_{\max}^{(t)}$ are the fitness of the worst and the best positions at the tth environment, respectively, T is the number of environments, ϑ is the change frequency, c is the fitness evaluation counter for each environment, and $\vec{x}^{*((t-1)\vartheta+c)}$ is the best found position at the cth fitness evaluation in the tth environment. As can be seen, to calculate E_A, the fitness value of the worst position in each environment must be known, which is not available even in most artificial DOP benchmark generators. In [122], an accuracy measure is proposed that does not use the fitness value of the worst position in each environment.

2) Efficiency based performance indicators: Among the existing efficiency based performance indicators, the distance based ones are the most commonly used. *Distance to optimum* (D_O) based performance indicators work according to the distance between the closest found solution to the global optimum. Note that, the distance can be calculated by $L2$-norm [2] (i.e., Euclidean distance) or infinity-norm [27]. To use these performance indicators, the optimum position must be known. D_O can be calculated as the average distance between the closest found position to the global optimum over all fitness evaluations, iterations, or at the end of each environment [129]. D_O that measures the tracking efficiency of DOAs over all fitness evaluations is formulated as:

$$D_O = \frac{1}{T\vartheta}\sum_{t=1}^{T}\sum_{c=1}^{\vartheta}\left\|\vec{x}^{\star(t)} - \vec{x}^{\circ((t-1)\vartheta+c)}\right\|, \qquad (\mathrm{P_B}\text{-}26)$$

where $\vec{x}^{\circ((t-1)\vartheta+c)}$ is the closest found position to the global optimum at the cth fitness evaluation in the tth environment. It is shown in [66] that there is no relation between D_O values and fitness/error based performance indicators (e.g., E_O and E_{BBC}). For instance, consider the case where a solution is fit, however, it resides on a peak that is far away from the global optimum. Although this solution is promising based on fitness/error based performance indicators, it is considered undesirable when D_O is used due to its long distance to the global optimum. Note that while D_O works based on the closest found solution to the optimum, most of DOAs work based on fitness. Hence, analyzing the behavior of these DOAs using such an indicator may not be accurate.

In [7], reaching a predefined distance to the optimum is used to evaluate the performance of DOAs. Besides, the average number of iterations needed to reach a predefined error after each environmental change is used as the performance indicator in [10]. In [122], a fitness adaptation speed measure is used that determines how quickly a DOA can reach its highest performance in each environment. Moreover, to measure the efficiency of DOAs in locating and covering peaks, a peak coverage measure is proposed in [193], which counts the number of located and covered peaks by DOAs. Similarly, a peak coverage rate is used in [3].

B. Plots

Plots of errors and fitness values are widely used in the DOP literature to facilitate analyzing and comparing the performance of DOAs. Figure $\mathrm{P_B}$-3 shows four commonly used types of plots, called current error, current performance, offline error, and offline performance. To draw the current error and offline error plots, we need to know the fitness value of the global optimum in each environment. The current performance plot shows the fitness value of the best found position at each fitness evaluation. Therefore, it illustrates the convergence behavior of DOAs in each environment, and also shows the fitness drop after each environmental change. To draw this plot, the information of the global optimum is not needed, which makes it suitable for more realistic circumstances where this information is not available. However, since the value of the optimum in each environment is unknown, the premature convergence cannot be recognized from such a plot. On the other hand, offline performance plot shows the value of offline performance (P_O) at each fitness evaluation, which is the average value of all previous *current performance* values.

Unlike current and offline performance plots that do not need the information of the global optimum, illustrating current and offline error plots need this information. Therefore, the global optimum value must be known in each environment, which makes current and offline error plots unsuitable for the situation where this information is not available, including real-world problems. Current error plot shows the error of the best found position at each fitness evaluation. This plot illustrates most of the information that current performance plot provides. In addition, the capability of algorithms in finding the global optimum can be observed from the current error plots. Offline error plot demonstrates the offline error over all fitness evaluations whose value at each fitness evaluation is the mean value of the previous current error values.

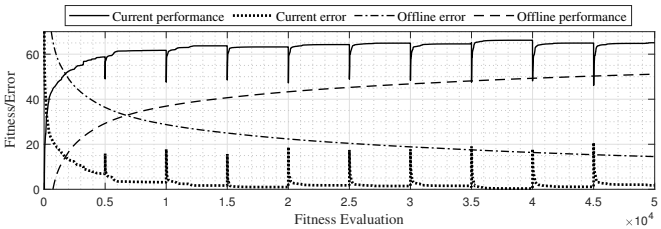


Fig. $\mathrm{P_B}$-3. An example of four commonly used plots in the DOP literature. These plots are obtained by FTmPSO [78] on MPB with the default settings of Scenario 2 and 10 environments [178]. The plots are obtained by averaging the results of 31 independent runs.

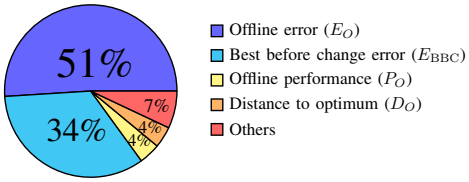


Fig. $\mathrm{P_B}$-4. Usage percentages of different performance indicators in the literature.

C. Discussion on performance analysis methods

Table $\mathrm{P_B}$-III lists the performance analysis methods used for comparison and analysis of DOAs together with the corresponding references that use them. As shown in this table, the majority of the DOP literature only use performance indicators to analyze the performance of DOAs. Figure $\mathrm{P_B}$-4 illustrates the usage percentages of different performance indicators in the DOP literature. Note that the performance indicators that have been rarely used in the DOP literature (less than 2%) are considered in the *others* group. As can be seen in this pie-chart, among all proposed performance indicators in the literature, four of them have been used frequently, namely E_O, E_{BBC}, P_O, and D_O. According to this pie-chart, more than 90% of the used performance indicators in the DOP literature are E_O, E_{BBC}, and P_O, which all are of fitness/error based type.

After discussing the most well-known performance indicators, the natural question is: *which performance indicator(s) should be used*? To answer this question, one thing to consider is the complexity of performance indicators. In fact, if the calculation of the performance indicator is complicated, analyzing the results would be complicated as well. Another factor to decide upon which indicator to use is availability of necessary information for the performance indicator. For example, when the global optimum information is not available, *error* based performance indicators cannot be used.

According to Table $\mathrm{P_B}$-III, efficiency based performance indicators are not popular as they do not focus on the most important aspect of DOAs which is the quality of the found solutions. In fact, even the most commonly used efficiency based indicators, which are the distance to optimum based ones, have been rarely used in the field. The reason is that, as mentioned before, the distance between the closest found solution to the optimum is not always related to the quality of solutions. Consequently, by only looking at the output of distance to optimum based indicators, we are generally unable to analyze the behavior of DOAs accurately.

Although E_{BBC} is very popular, it is not suitable for practical situations as the best results at the end of environments cannot be useful to show the performance of DOAs in real-world problems. As discussed before, the fitness of the deployed solutions in DOPs is important to measure the performance of DOAs. Therefore, E_{BBD} ($\mathrm{P_B}$-27) with different system tolerance thresholds (η) can be very useful to analyze the performance of DOAs in a more practical way. Note that, if η is set to $\vartheta - 1$, the obtained values by E_{BBD} will be equal to those of E_{BBC}. Additionally, in cases where the optimum information is not available, E_{BBD} can be changed from an error based performance indicator to a fitness based one as follows:

$$P_{\mathrm{BBD}} = \frac{1}{T}\sum_{t=1}^{T} f^{(t)}\left(\vec{x}^{*(t-1)*\vartheta+\eta}\right). \qquad (\mathrm{P_B}\text{-}27)$$

Another important consideration to have a fair comparison among DOAs using the performance indicators is to use the same environmental parameters for different DOAs. In fact, environmental changes in DOPs are usually dependent on several random number generators. Consequently, by using different random seed numbers, the landscape of the next environment can be different in terms of various aspects, especially its degree of difficulty. Therefore, comparing different DOAs on problem instances whose environments are different due to the various random seed values, can be biased significantly. To address the aforementioned issue, the environments must be generated and changed with the same sequence of random seed numbers for all comparing algorithms. This makes the obtained results by the performance indicators, especially by the fitness based ones that directly use the average fitness values (e.g., P_O), unbiased.

Finally, the plots of current error/performance should be presented by researchers since they demonstrate several important information including the convergence behavior of DOAs and fitness drops, which cannot be captured by the performance indicators.

$\mathrm{P_B}$-IV. Optimization components

Evolutionary and swarm intelligence algorithms (EAs and SIs) are originally designed for solving static optimization problems. Hence, they cannot be applied for optimization of a DOP directly due to the challenges of DOPs, such as diversity loss. In order to make these algorithms suitable for the purpose, a DOA uses them together with some other necessary components. As already discussed in the first part of this survey, a DOA is usually made by assembling several components (see § $\mathrm{P_A}$-III) to address the challenges of DOPs.

EAs and SIs are the most efficient tools to be embedded in DOA frameworks to tackle DOPs [182]. In 1966, Fogel et al. applied an EA to DOPs for the first time [195]. Later on, in 2000, Carlisle and Dozier initiated the use of SIs to DOPs [7]. The studies of applying EAs and SIs in DOPs are known as evolutionary dynamic optimization (EDO) [182] and swarm intelligence dynamic optimization (SIDO) [196], respectively. Figure $\mathrm{P_B}$-5 shows the popularity of different

TABLE P_B-III
USED PERFORMANCE ANALYSIS METHODS IN THE DOP LITERATURE AND CORRESPONDING REFERENCES.

	Measure	References
Plots	Current performance	[19], [43], [64], [122], [132], [146], [167], [175]
	Current error	[8], [11]–[13], [16], [17], [22], [96], [122], [145], [176]
	Offline performance over time	[30], [62]
	Offline error over time	[12], [57], [62]
	Others	[125], [177]
Performance indicator	Best before change error	[26], [27], [29], [41], [42], [51], [63], [72], [74], [75], [78], [82], [83], [87], [94], [97], [100], [106]–[108], [113]–[118], [121]–[123], [130], [134], [135], [137], [142]–[144], [147]–[162], [164]–[166], [168], [170], [172], [174], [194]
	Offline performance	[18], [28], [35], [43], [62], [122], [126]
	Offline error (evaluation based)	[15], [18], [22]–[25], [27], [31]–[34], [36]–[40], [44]–[50], [52]–[57], [59]–[63], [67], [68], [71], [73], [76], [78], [80], [84]–[87], [89]–[94], [96], [98], [99], [101], [103], [105], [106], [108]–[113], [117]–[122], [127], [136], [138], [140]–[143], [166]
	Offline error (iteration based)	[26], [29], [58], [69], [70], [79], [81], [102], [104], [158], [163], [169]
	Distance based performance indicators	[2], [9], [14], [27], [88], [116], [129]
	Others	[3], [7], [10], [21], [35], [57], [65], [65], [77], [77], [122], [128]

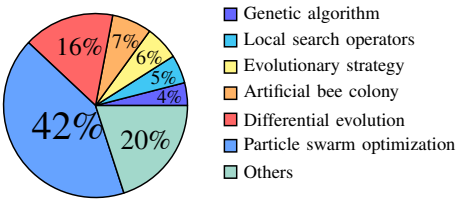


Fig. P_B-5. Usage percentages of different static optimization algorithms, including their modified versions, in the DOAs.

TABLE P_B-IV
STATIC OPTIMIZATION ALGORITHMS USED AS THE OPTIMIZATION COMPONENT IN DOAS.

Static optimization algorithm	Corresponding DOA references using the static optimization algorithm
Particle swarm optimization [197]	[3], [7], [8], [10]–[12], [18], [23], [25], [27], [31], [32], [34], [36], [37], [39]–[44], [46]–[49], [51]–[55], [57], [60], [62], [64], [68]–[72], [74], [77], [78], [80], [86]–[88], [94], [97], [100], [106]–[108], [116], [118], [121], [123], [125]–[131], [134], [135], [137], [142]–[145], [149], [162], [166], [168], [169], [172], [174], [199]–[208]
Differential evolution [209]	[33], [38], [45], [56], [61], [63], [74], [75], [80], [81], [106], [108], [114], [115], [117], [134], [140], [141], [143], [145], [146], [148], [154]–[159], [161], [164], [165], [168], [170]
Artificial bee colony [210]	[22], [24], [96], [99], [102], [104], [109], [110], [151], [152], [163], [176], [177]
Local search operators [211]	[36], [70]–[72], [80], [87], [90], [101]
Genetic algorithm [212]	[19], [29], [76], [132], [134], [175]
Evolutionary strategy[a][213]	[2], [9], [13], [14], [17], [20], [21], [145], [150], [171], [214]
Evolutionary programming [215]	[16], [41]
Evolutionary algorithm [216]	[15], [26], [28], [30], [217], [218]
Firefly algorithm [219]	[67], [85], [92], [120], [136]
Bacterial foraging algorithm [220]	[35], [65], [79], [167]
Ant colony optimization[b][221]	[147], [154]
Artificial fish swarm algorithm [222]	[73], [84], [91]
Harmony search algorithm [223]	[83], [111]
Cuckoo search algorithm [224]	[89], [93], [103]
Artificial immune system [225]	[112], [113], [139], [164], [226], [227]
Others	[59], [82], [98], [105], [119]

[a] Including CMA-ES [228].
[b] Continuous version [221].

static optimization algorithms that have been embedded in DOAs as the optimization component since 1999. According to this pie-chart, particle swarm optimization (PSO) [197] is the most popular optimization tool in the field of DOPs. The main reasons are high convergence speed, and efficient exploration and exploitation capabilities of PSO [198] that make it suitable to be used in DOA frameworks.

While the main optimization component in DOAs is usually an EA or SI, local search operators mostly have been used in DOAs to improve the exploitation process around the best found position in each environment (see § P_A-III-E2e). However, in [90], [101], the local search operators play the role of the main optimization components and perform locating and tracking peaks. Note that, using a local search operator as the main optimization tool can potentially lead to immature/early convergence. However, as the peaks of the landscapes of the most commonly used DOP benchmarks are mostly unimodal and easy to optimize (see § P_B-II-A1), using such simple optimization component can be shown to be efficient.

There have been many attempts to modify and adapt EAs and SIs to be used by DOAs [7], [38], [44], [73], [119], [152]. To this end, an existing DOA framework is chosen, and the considered EA or SI is embedded in it. In DynDE [33], DE is embedded in the introduced DOA framework in [32]. DynDE is the first EDO based on DE. Charged PSO [229], which is originally proposed in the context of artificial improvised music, is used in the presented DOAs in [12], [31], [32], [131]. Charged PSO uses some collision avoidance rules, which make it capable of maintaining diversity. In [44], [77], cooperative PSO [230], which is a cooperative coevolutionary method, is embedded in the DOA framework from [32]. In [162], orthogonal learning PSO [231] is used as the optimization component in the DOA framework from [15]. Moreover, in [36], guaranteed convergence PSO [232], that uses some additional movement rules to perform local search around the best found position, is used in the DOA framework from [128]. Crowding DE [233], which is an algorithm that is originally designed for detecting multiple peaks in static multimodal optimization problems, has been used to carry

out exploration and locating multiple peaks in [38], [52], [81]. Another static multimodal optimization method is vector-based PSO [234], which is used to cover multiple moving peaks in [3]. Heterogeneous DE [235] is another algorithm that is originally designed for global optimization in static problems that has been adapted to solve DOPs in [170]. Different PSOs with various neighborhood topologies, such as the closest individual [174], ring [58], Von Neumann [125], and hierarchical [18], [236], have been used in DOAs.

In addition, some researchers modify the EAs and SIs to adapt them to DOPs. In [126], a macro-mutation operator is added to PSO to control its population diversity. An oscillating inertia weight is designed for PSO in [88] to control the trade-off between exploration and exploitation capabilities in DOPs. A PSO with a fuzzy social-only model is proposed in [87], where the exploration capabilities of the PSO is improved. In [158], to maintain the population diversity, a DE with double mutation strategy and modified scaling factors is used.

A list of the static optimization algorithms that have been used in DOAs and the corresponding references is provided in Table $\mathrm{P_B}$-IV. This table classifies DOAs based on their optimization components and can be useful for the researchers who are interested in modifying and adapting different static optimization algorithms for DOA frameworks.

A. Discussion on the used optimizers in DOAs

A considerable portion of DOP papers focus on modification of static optimization algorithms to be adapted to an existing DOA framework. We have reviewed some works that focus on modifying and adapting the structures of such static optimization algorithms to be used in DOA frameworks. Designing components of DOAs to tackle DOP challenges is another major topic covered in the literature, which is independent of the used optimization tool. In such works, after designing a DOA framework, which is developed by combining several components (see § $\mathrm{P_A}$-III), a static optimization algorithm(s) is chosen to be embedded in the framework as the optimization component. According to our readings, PSO (e.g., with global star neighborhood topology and constriction factor [10]) and DE (e.g., DE/best/2/bin [56]) are usually the best options for the optimization component and have been widely used in the literature. Note that the majority of DOA frameworks show their best performance when PSO is used as the optimization component [134], [143], [145], [180].

$\mathrm{P_B}$-V. Real-world DOPs

Changes and uncertainties are natural aspects of many real-world optimization problems, hence an optimization algorithm searching in such environments should be able to efficiently respond to such demands [182]. In this section, we review some real-world DOPs that have been solved using DOAs reviewed in § $\mathrm{P_A}$-III.

In [199], [200], DOAs are applied to train supervised feed-forward neural networks in dynamic classification problems with concept drift[4]. For pattern selection in the concept drift, a sliding window is used, and the change severity is adjusted using the values of the sliding window's step size. The severity of concept drift varies from gradual (slightly shifting the decision boundaries) to highly severe changes (randomly creating new decision boundaries) to generate different problem instances [238]. The environmental changes in this problem are visible (i.e., detectable [182]) and can be detected by observing the error values. Several PSO based DOAs, including RPSO [10] (single-population PSO with randomization after changes for tracking moving optimum), mCPSO, and mQSO [31], [32] (multi-population PSOs for tracking multiple moving promising regions with charged and quantum individuals, respectively), and two back propagation (BP) methods [239] are used for different problem instances with various data sets. On the one hand, the results show that PSO based DOAs outperform both BP methods in problem instances with relatively moderate change frequencies. On the other hand, PSOs are deficient in problem instances with high change frequencies due to the very limited available computational resources in each environment. In addition, both mCPSO and mQSO outperform RPSO in problem instances with gradual concept drift shifts where successive environments are rather similar, and the information from the previous environment is highly useful to accelerate the process of tracking the moving optimum. On the other hand, when the environmental changes are highly severe (abrupt), RPSO outperforms both mCPSO and mQSO since the information of the previous environments is significantly less useful for optimization in the next environment, and reinitializing a significant portion of the population is in fact more effective.

Kalita and Singh [207] use a DOA to optimize the hyper-parameters of a support vector machine[5] (SVM) in dynamic environments. Two cases of dynamics are investigated in this problem: when a stream of data is received gradually, and when data is received in batches at some points of time. In the first case, environmental changes are very smooth as new data arrives gradually over time. On the other hand, the environmental changes in the second case are more severe since receiving a batch of data causes a relatively severe environmental change. It is also shown that the knowledge of the previous global optimum is useful in this problem and can lead to acceleration of the optimization process after environmental changes. To tackle this problem, the authors applied a multi-population PSO where the main population is clustered into subpopulations using the niching approach presented in [234]. In addition, each subpopulation uses quantum and charged individuals [32] to maintain its local diversity over time. To address global diversity loss, the exclusion and anti-convergence [32] components are used in this DOA.

In another study [208], a cutting pattern recognition method is proposed for shearer in coal mining which is based on an SVM whose hyper-parameters are optimized by a PSO-based DOA. In this problem, the training dataset is updated over time. A chaos-based operator is used in the DOA to increase diversity. In this method, if the population has converged or stagnated, a predefined number of individuals are relocated

[4]Drifting concept in classification implies changes in the decision boundaries which separate different classes [237].

[5]This problem is also called SVM model selection.

using the chaos-based operator.

In [206], a DOA is used for training a neural network time series forecaster. The time series' data generating process is non-stationary, and a sliding window defines the training data over time. This sliding window shifts on the time series data when the environmental change happens. The introduced cooperative quantum PSO in [77] is used as the DOA to tackle this problem.

Jin et al. [217] applied a DOA to optimize the adaptive farming strategies, where the problem is to maximize the income by systematically choosing mixed grazing strategies. The problem is a simulation optimization based on UK farming data and subsidy policy. It has three dynamics: the biological dynamics of grouse populations, government subsidies, and the price of farming products. The environmental changes in this problem are not abrupt and there are similarities between successive environments. A single population EA is used as the optimization method whose mutation rate is considered relatively high to maintain its diversity over time.

The control strategy parameters for a distribution static compensator (DSTATCOM) are optimized by a DOA in [226], [227]. In this problem, the main objective is to control the parameters of the DSTATCOM in order to minimize the negative effects of the pulsed loads in an all-electric ship power system. A real-time digital simulator is used as the objective function which is capable of simulating the dynamics of the system. A PSO is used to find an initial static solution in an initial offline phase. Thereafter, the deviation of voltage is kept minimum during the real-time process using an artificial immune system (AIS).

The odor source localization problem is solved by DOAs in [201]. The main objective of this problem is to locate the source of chemical odor using mobile robots. The problem space is highly irregular due to obstacle-filled environments, and very dynamic due to the wind (changing force and direction), turbulence in the distribution of odor molecules, and diffusion of odor. A single-population PSO is used to solve this problem where there is one source of odor in the environment. This algorithm uses a change detection method by monitoring the fitness value of the best found position. If this value does not improve over a predefined number of iterations, it is assumed that a change has occurred. After detecting an environmental change, the algorithm increases the global diversity by spreading robots in different directions. In addition, the local diversity is maintained over time using charged individuals [12]. This work is expanded in [202] to locate *multiple* odor sources using a multi-population PSO method [240]. By considering multiple odor sources, the problem becomes dynamic multimodal optimization and the DOA needs to locate and track multiple moving optima.

In [203], [204], DOAs are applied to the dynamic economic dispatch problem to minimize the operational cost in an electrical power system. In [203], an adaptive single-population PSO is used. In this variant of PSO, each particle has its own inertia weight, which is defined based on its fitness rank in the population. While superior particles have lower inertia weight to increase their exploitation capability, inferior particles have larger inertia weights that result in maintaining their high velocity to preserve diversity. Moreover, if the best found position does not improve noticeably after a specific number of iterations, a predefined number of individuals are randomized across the search space to increase the global diversity. In [204], the velocity update rule of PSO is modified to dynamically adjust the trade-off between exploitation and exploration capabilities. In this method, the velocity is decreased when the best found position does not improve significantly over a predefined number of iterations. On the contrary, the velocity is increased when the best found position significantly improves over the same number of iterations. Moreover, a chaotic operator is added to PSO to increase the global diversity.

Liu et al. [214], [218] apply a DOA to a contaminant source identification problem in water distribution networks, for which the dynamics are the result of the partial information about the problem. Some sensors are used to gather information from the environment and whenever the information is updated, the search space is changed. This problem has a *quick recovery* property [190], where there is a deadline to choose a new solution (a new solution must be chosen before new observations are received). A multi-population EA [241] is used to locate several good solutions (peaks) and track them over time. In this method, an exclusion component is used to avoid overlapping subpopulations. It is shown that any of the covered promising regions in this problem may contain the global optimum as time goes by.

A. Discussion on real-world applications

Despite the importance of the real-world DOPs, the majority of the literature is focused on the artificial DOP benchmarks and little attention has been given to application of DOAs for solving real-world DOPs. The rare studies focused on real-world application of DOAs mostly investigate the performance of simple DOAs with limited numbers of dynamic components. Hence, the effectiveness of most DOAs in optimizing real-world DOPs are not entirely clear.

$\mathrm{P_B}$-VI. Potential future research directions

Although tracking the moving optimum in unconstrained continuous single-objective DOPs has been a hot research topic over the past two decades with a rich literature, there is still a considerable gap between academic research and practical applications. In fact, most of the developed DOAs have only been tested on artificial functions, i.e., DOP benchmarks. To the best of our knowledge, the majority of the components developed for DOAs (see § $\mathrm{P_A}$-III) are tailored for such artificial DOP benchmarks, in particular, for moving peaks baseline functions (see § $\mathrm{P_B}$-II-A1). Although a few DOAs have been applied to some real-world DOPs (see § $\mathrm{P_B}$-V), the effectiveness of most DOAs for different real-world DOPs with various characteristics and challenges is not yet entirely clear. Many real-world DOPs with specific characteristics have not been adequately investigated in the literature, such as DOPs with limitations in frequent switching solutions [189], constrained DOPs [190], [242], high-dimensional DOPs [145], continuously changing over time DOPs, and time-linkage

DOPs [243]. Based on the current research status of the field, we point out the following potential future research directions to dwindle the gap between academic research and the real-world applications.

1) Real-world DOPs: Many critical real-world optimization problems are dynamic [190]. However, the majority of the literature is focused on artificial benchmark problems and only a few works address real-world DOPs (see § $\mathrm{P_B}$-V). One extremely important research direction is to formulate objective functions for real-world DOPs and design new DOAs to solve them. Using such formulated objective functions, we also can examine the effectiveness of the existing DOAs in solving realistic problems. An important group of real-world DOPs that have been ignored in the DOP literature are dynamic covering location problems (DCLPs) [244]. DCLPs cover a considerable number of critical real-world applications, such as disaster relief operation [245], large-scale fire management [246], [247], product positioning [248], and crowd management/control [249].

2) Analyzing the performance and compatibility of components: In a DOA, a combination of several components are used (see § $\mathrm{P_A}$-III). When comparing different DOAs, the contribution of each component to the performance of each DOA is unclear. In addition, the compatibility of these components has been rarely investigated. For example, we yet do not know how much a particular global diversity control component is compatible with a specific population division and management component. Besides, the effectiveness of components in different classes of DOPs, such as drifting DOPs with a high change frequency [95], is not entirely clear yet. Therefore, a comprehensive analysis of the performance and compatibility of different components in different classes of DOPs is an important future direction.

3) Automatic parameter tuning: Although various components are developed for DOAs, those with adaptive parameter tuning are rare. The majority of components work on the basis of some constant thresholds which need to be tuned for every problem. The performance of such components is highly dependent on the values decided for these thresholds. Besides, the optimal values of these thresholds depend on the characteristics of the current environment, hence, may change over time. This shows the need for components with automatic parameter tuning [250].

4) Hyperheuristics: In § $\mathrm{P_A}$-III, we discussed the strengths and weaknesses of different components of DOAs in solving DOPs. An important point here is that if the characteristics of a DOP change over time (e.g., it becomes unpredictable while it used to be predictable in the previous environments), choosing a proper set of components becomes difficult or sometimes even impossible. In such DOPs, hyperheuristic approaches can be used in DOAs to choose the best set of components depending on the current status of the problem [251], [252]. Hyperheuristic approaches can also be used for selecting the optimization update rules (e.g., update rules of PSO or DE) [253]. Using hyperheuristic approaches to choose the most proper set of components and/or metaheuristics for DOPs whose characteristics and challenges change over time is a promising future direction.

5) Computational resource allocation: In the literature, little attention has been given to the management of computational resources and assigning them to subpopulations. Most existing multi-population DOAs use the simple Round Robin/parallel method for running subpopulations in each iteration. However, using a systematic computational resource allocation method seems essential for DOPs with limited available computational resources in each environment. This need becomes more important when different covered regions by subpopulations may go under different change severity, hence, subpopulations covering them need different amount of computational resources to fulfill the tracking task. Developing new resource allocation methods that take the importance, role, progress, and rank of subpopulations into consideration, is a potential future work.

6) Source codes: Looking at the DOP literature, the results reported for a given DOA can sometimes be significantly different. This issue occurs as the source codes of many DOAs and benchmark problems are not available. Since DOAs are usually very complex algorithms, re-implementing them is usually error prone which leads to reporting false results. Therefore, providing a collection of source codes of the state-of-the-art and popular DOAs and benchmark problems is necessary. Providing a proper set of source codes as a platform implemented in popular programming languages, similar to [254], can be very useful for fair and accurate comparisons in the future.

7) Extension to other related types of DOPs:

a) Robust optimization over time: The majority of the works in the field are focused on the tracking of a moving optimum, which does not meet the needs of some real-world problems. Tracking a moving optimum is suitable for DOPs in which changing solutions is not costly [255]–[257], while in many real-world DOPs with high switching costs, it is desirable to keep the current solution while it is of an acceptable quality. Such solutions are called *robust solutions* which can maintain their qualities after a number of environmental changes (at least one environmental change). The process of finding robust solutions is called robust optimization over time (ROOT) [189]. ROOT can be tackled effectively by extending DOAs that have been originally designed for tracking moving optima [180]. Hence, extending DOAs by adding new components to tackle ROOT, designing benchmark problems which generate areas with higher robustness over time, and solving real-world problems with high switching costs using ROOT algorithms are all promising future research directions.

b) Constrained DOPs: Despite the importance of constrained DOPs, little attention has been given to them in the literature. In fact, many real-world DOPs are subjected to constraints. Constrained DOPs are difficult to optimize since their objective function and constraints may change over time. These problems can become more challenging when there are multiple disjointed moving feasible regions [242]. We can adapt DOAs to optimize constrained DOPs by equipping them with constraint handling mechanisms, such as penalty methods [242] and Deb's rules for selections and comparisons [258]. Thus, by using such mechanisms, DOAs will

also be capable of tracking moving optimum in constrained DOPs. However, tracking moving optimum by just using constraint handling mechanisms without additional components that consider multiple moving feasible regions, will not be effective [187]. Hence, an important future work direction is to extend and design DOAs to handle constrained DOPs.

c) Time-linkage DOPs: Although many real-world DOPs have *time-linkage* property [178], [190], only few works have investigated DOPs with such a feature. In time-linkage DOPs, the chosen solution in one environment influences the environment encountered in the future. To tackle such problems, DOAs usually need to use prediction/estimation components to consider the future impact of the current solutions [190]. Considering time-linkage DOPs is a potential future work.

d) Large-scale DOPs: Many real-world DOPs have a large number of variables, which causes *scalability issues* and the *curse of dimensionality* [180]. Only a few works have investigated large-scale DOPs, such as the work done in [145], where a cooperative coevolutionary algorithm is proposed on the basis of a multi-population DOA to effectively solve large-scale DOPs. Large-scale DOPs are difficult problems since the available computational resources in each environment can be very limited to perform optimum tracking in high-dimensional search space. This is even more critical for multi-population DOAs which track multiple moving promising regions using several subpopulations that need a lot of computational resources. In partially separable large-scale DOPs, the overall number of peaks can exponentially grow [180] which makes them even more challenging. Hence, designing DOAs to solve large-scale DOPs is an important future direction.

e) DOPs with a huge number of promising regions (peaks): Search space of some real-world DOPs may contain a huge number of promising regions. In the DOP literature, the effectiveness of DOAs has rarely been investigated on such DOPs (number of peaks in the problem instances has rarely exceeded 200). Since the population size is limited in DOAs, covering and tracking too many moving optima will be almost impossible. Therefore, performance of many existing DOAs deteriorates in tackling these DOPs. Designing population division and management and global diversity control components to address the challenges posed by DOPs with a huge number of promising regions is an interesting future work.

f) DOPs that continuously change over time: Almost all DOAs are designed for DOPs with environmental changes happening over discrete time. However, there are real-world DOPs whose environments continuously change over time or their change frequencies are extremely high. The majority of the existing DOAs are change-dependent (change reaction based), hence, are incapable of tackling such DOPs efficiently. Targeting these DOPs involves many new challenges that need further investigations.

g) Dynamic multi-objective optimization problems: Dynamic multi-objective optimization problems (DMOPs) have several conflicting objective functions that must be considered simultaneously. The works on DMOPs mostly focus on finding Pareto optimal set (POS) for each environment [259], [260]. In these works, the found POS from the previous environment(s) is usually used to accelerate the process of finding the new POS after each environmental change [261]. As finding POS is computationally more expensive than finding a single solution (i.e., the global optimum in single-objective), algorithms developed for DMOPs often struggle with very limited available computational resources in each environment [261]. Developing components for accelerating the process of tracking the POS in DMOPs is another potential future direction.

8) Theoretical investigations: In the DOP literature, the performance of DOAs have been usually investigated empirically, and little attention has been given to theoretical studies [182], [262]. Due to the importance of theoretical studies in understanding the behavior of DOAs and their strength and weakness in tackling DOPs with different characteristics, more attention should be dedicated to such investigations.

P_B-VII. Conclusion

In this two-part survey, we reviewed the field of evolutionary continuous dynamic optimization over the last two decades. Part A of this survey was focused on the components of dynamic optimization algorithms (DOAs). In Part B, we provided a detailed review of dynamic optimization problem (DOP) benchmarks, DOP performance indicators and plots used for analyzing the performance of DOAs, static optimization algorithms used as optimization components in DOAs, and the real-world applications of some DOAs reviewed in Part A. We also discussed the current shortcomings of the field and provided some important potential future research directions.

References

[1] D. Yazdani, R. Cheng, D. Yazdani, J. Branke, Y. Jin, and X. Yao, "A survey of evolutionary continuous dynamic optimization over two decades – part A," *IEEE Transactions on Evolutionary Computation*, 2021.

[2] K. Weicker and N. Weicker, "On evolution strategy optimization in dynamic environments," in *IEEE Congr. Evol. Comput.*, vol. 3. IEEE, 1999.

[3] I. Schoeman and A. Engelbrecht, "Niching for dynamic environments using particle swarm optimization," in *Simulated Evolution and Learning*, T.-D. Wang et al., Ed. Springer Berlin Heidelberg, 2006, pp. 134–141.

[4] J. J. Liang, P. N. Suganthan, and K. Deb, "Novel composition test functions for numerical global optimization," in *Swarm Intelligence Symposium*. IEEE, 2005, pp. 68–75.

[5] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari, "Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization," Nanyang Technological University, Tech. Rep., 2005.

[6] C. Li, S. Yang, T. T. Nguyen, E. L. Yu, X. Yao, Y. Jin, H.-G. Beyer, and P. N. Suganthan, "Benchmark generator for cec'2009 competition on dynamic optimization," Center for Computational Intelligence, Tech. Rep., 2008.

[7] A. Carlisle and G. Dozier, "Adapting particle swarm optimization to dynamic environments," in *International conference on Artificial Intelligence*, 2000, pp. 429–434.

[8] R. C. Eberhart and Y. Shi, "Tracking and optimizing dynamic systems with particle swarms," in *IEEE Congr. Evol. Comput.*, vol. 1. IEEE, 2001, pp. 94–100.

[9] D. V. Arnold and H.-G. Beyer, "Random dynamics optimum tracking with evolution strategies," in *Parallel Problem Solving from Nature — PPSN VII*, J. J. M. Guervós et al., Ed. Springer Berlin Heidelberg, 2002, pp. 3–12.

[10] X. Hu and R. C. Eberhart, "Adaptive particle swarm optimization: detection and response to dynamic systems," in *IEEE Congr. Evol. Comput.*, vol. 2. IEEE, 2002, pp. 1666–1670.

[11] A. Carlisle and G. Dozler, "Tracking changing extrema with adaptive particle swarm optimizer," in *World Automation Congress*, vol. 13. IEEE, 2002, pp. 265–270.

[12] T. M. Blackwell and P. J. Bentley, "Dynamic search with charged swarms," in *Genet. Evol. Comput. Conf.* Morgan Kaufmann Publishers Inc., 2002, pp. 19–26.

[13] A. Boumaza, "Learning environment dynamics from self-adaptation: A preliminary investigation," in *Genet. Evol. Comput. Conf.* ACM, 2005, pp. 48–54.

[14] D. V. Arnold and H.-G. Beyer, "Optimum tracking with evolution strategies," *Evololutionary computation*, vol. 14, no. 3, pp. 291–308, 2006.

[15] Y. G. Woldesenbet and G. G. Yen, "Dynamic evolutionary algorithm with variable relocation," *IEEE Trans. Evol. Comput.*, vol. 13, no. 3, pp. 500–513, 2009.

[16] P. Angeline, "Tracking extrema in dynamic environments," in *Evolutionary Programming VI*, P. Angeline et al., Ed. Springer Lecture Notes in Computer Science, 1997, vol. 1213, pp. 335–345.

[17] T. Back, "On the behavior of evolutionary algorithms in dynamic environments," in *IEEE Congr. Evol. Comput.* IEEE, 1998, pp. 446–451.

[18] S. Janson and M. Middendorf, "A hierarchical particle swarm optimizer for dynamic optimization problems," in *Applications of Evolutionary Computing*, G. R. Raidl et al., Ed. Springer Berlin Heidelberg, 2004, pp. 513–524.

[19] F. Oppacher and M. Wineberg, "The shifting balance genetic algorithm: Improving the ga in a dynamic environment," in *Genet. Evol. Comput. Conf.*, vol. 1. ACM, 1999, pp. 504–510.

[20] L. Schonemann, "The impact of population sizes and diversity on the adaptability of evolution strategies in dynamic environments," in *IEEE Congr. Evol. Comput.*, vol. 2. IEEE, 2004, pp. 1270–1277.

[21] L. Schönemann, *Evolution Strategies in Dynamic Environments*. Springer Berlin Heidelberg, 2007, pp. 51–77.

[22] M. Kojima, H. Nakano, and A. Miyauchi, "An artificial bee colony algorithm for solving dynamic optimization problems," in *IEEE Congr. Evol. Comput.* IEEE, 2013, pp. 2398–2405.

[23] I. G. del Amo, D. A. Pelta, and J. R. González, "Using heuristic rules to enhance a multiswarm pso for dynamic environments," in *IEEE Congr. Evol. Comput.* IEEE, 2010, pp. 1–8.

[24] H. Nakano, M. Kojima, and A. Miyauchi, "An artificial bee colony algorithm with a memory scheme for dynamic optimization problems," in *IEEE Congr. Evol. Comput.* IEEE, 2015, pp. 2657–2663.

[25] W. Du and B. Li, "Multi-strategy ensemble particle swarm optimization for dynamic optimization," *Inf. Sci.*, vol. 178, no. 15, pp. 3096 – 3109, 2008.

[26] A. Meier and O. Kramer, "Prediction with recurrent neural networks in evolutionary dynamic optimization," in *Applications of Evolutionary Computation*, K. Sim and P. Kaufmann, Eds. Springer International Publishing, 2018, pp. 848–863.

[27] J. Karimi, H. Nobahari, and S. Pourtakdoust, "A new hybrid approach for dynamic continuous optimization problems," *Appl. Soft Comput.*, vol. 12, no. 3, pp. 1158 – 1167, 2012.

[28] J. Branke, "Memory enhanced evolutionary algorithms for changing optimization problems," in *IEEE Congr. Evol. Comput.*, vol. 3. IEEE, 1999, pp. 1875–1882.

[29] K. Trojanowski and Z. Michalewicz, "Searching for optima in non-stationary environments," in *IEEE Congr. Evol. Comput.*, vol. 3, 1999, pp. 1843–1850.

[30] J. Branke, T. Kaussler, C. Schmidt, and H. Schmeck, "A multi-population approach to dynamic optimization problems," in *Evolutionary Design and Manufacture*. Springer, 2000, pp. 299–307.

[31] T. Blackwell and J. Branke, "Multi-swarm optimization in dynamic environments," in *Applications of Evolutionary Computing*, G. R. Raidl et al., Ed. Lecture Notes in Computer Science, 2004, vol. 3005, pp. 489–500.

[32] ——, "Multiswarms, exclusion, and anti-convergence in dynamic environments," *IEEE Trans. Evol. Comput.*, vol. 10, no. 4, pp. 459–472, 2006.

[33] R. Mendes and A. S. Mohais, "DynDE: a differential evolution for dynamic optimization problems," in *IEEE Congr. Evol. Comput.*, vol. 3. IEEE, 2005, pp. 2808–2815.

[34] X. Li, J. Branke, and T. Blackwell, "Particle swarm with speciation and adaptation in a dynamic environment," in *Genet. Evol. Comput. Conf.* ACM, 2006, pp. 51–58.

[35] W. J. Tang, Q. H. Wu, and J. R. Saunders, "Bacterial foraging algorithm for dynamic environments," in *Conference on Evolutionary Computation*. IEEE, 2006, pp. 1324–1330.

[36] S. Bird and X. Li, "Using regression to improve local convergence," in *IEEE Congr. Evol. Comput.* IEEE, 2007, pp. 592–599.

[37] T. Blackwell, *Particle Swarm Optimization in Dynamic Environments*. Springer Berlin Heidelberg, 2007, pp. 29–49.

[38] R. I. Lung and D. Dumitrescu, "A collaborative model for tracking optima in dynamic environments," in *IEEE Congr. Evol. Comput.* IEEE, 2007, pp. 564–567.

[39] H. Wang, D. Wang, and S. Yang, "Triggered memory-based swarm optimization in dynamic environments," in *Applications of Evolutionary Computing*, M. Giacobini, Ed. Springer Berlin Heidelberg, 2007, pp. 637–646.

[40] T. Blackwell, J. Branke, and X. Li, "Particle swarms for dynamic optimization problems," in *Swarm Intelligence: Introduction and Applications*, C. Blum and D. Merkle, Eds. Springer Lecture Notes in Computer Science, 2008, pp. 193–217.

[41] C. Li and S. Yang, "Fast multi-swarm optimization for dynamic optimization problems," in *International Conference on Natural Computation*, vol. 7. IEEE, 2008, pp. 624–628.

[42] L. Liu, D. Wang, and S. Yang, "Compound particle swarm optimization in dynamic environments," in *Applications of Evolutionary Computing*, M. Giacobini et al., Ed. Springer Berlin Heidelberg, 2008, pp. 616–625.

[43] H. Wang, N. Wang, and D. Wang, "Multi-swarm optimization algorithm for dynamic optimization problems using forking," in *Control and Decision Conference*. IEEE, 2008, pp. 2415–2419.

[44] A. Rakitianskaia and A. P. Engelbrecht, "Cooperative charged particle swarm optimiser," in *IEEE Congr. Evol. Comput.* IEEE, 2008, pp. 933–939.

[45] M. C. du Plessis and A. P. Engelbrecht, "Improved differential evolution for dynamic optimization problems," in *IEEE Congr. Evol. Comput.* IEEE, 2008, pp. 229–234.

[46] A. B. Hashemi and M. R. Meybodi, "Cellular pso: A pso for dynamic environments," in *Advances in Computation and Intelligence*, Z. Cai et al., Ed. Springer Berlin Heidelberg, 2009, pp. 422–433.

[47] A. B. Hashemi and M. R. Meybodi, "A multi-role cellular pso for dynamic environments," in *International CSI Computer Conference*. IEEE, 2009, pp. 412–417.

[48] K. Trojanowski, "Properties of quantum particles in multi-swarms for dynamic optimization," *Fundamenta Informaticae*, vol. 95, no. 2-3, pp. 349–380, 2009.

[49] P. Novoa, D. A. Pelta, C. Cruz, and I. G. del Amo, "Controlling particle trajectories in a multi-swarm approach for dynamic optimization problems," in *Methods and Models in Artificial and Natural Computation. A Homage to Professor Mira's Scientific Legacy*, J. Mira et al., Ed. Springer Berlin Heidelberg, 2009, pp. 285–294.

[50] D. Pelta, C. Cruz, and J. L. Verdegay, "Simple control rules in a cooperative system for dynamic optimisation problems," *International Journal of General Systems*, vol. 38, no. 7, pp. 701–717, 2009.

[51] S. Yang and C. Li, "A clustering particle swarm optimizer for locating and tracking multiple optima in dynamic environments," *IEEE Trans. Evol. Comput.*, vol. 14, no. 6, pp. 959–974, 2010.

[52] R. I. Lung and D. Dumitrescu, "Evolutionary swarm cooperative optimization in dynamic environments," *Natural Computing*, vol. 9, no. 1, pp. 83–94, 2010.

[53] M. Kamosi, A. B. Hashemi, and M. R. Meybodi, "A new particle swarm optimization algorithm for dynamic environments," in *Swarm, Evolutionary, and Memetic Computing*, B. K. Panigrahi et al., Ed. Springer Berlin Heidelberg, 2010, pp. 129–138.

[54] M. Kamosi, A. B. Hashemi, and M. R. Meybodi, "A hibernating multi-swarm optimization algorithm for dynamic environments," in *Nature and Biologically Inspired Computing*. IEEE, 2010, pp. 363–369.

[55] L. Liu, S. Yang, and D. Wang, "Particle swarm optimization with composite particles in dynamic environments," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 40, no. 6, pp. 1634–1648, 2010.

[56] M. C. du Plessis and A. P. Engelbrecht, "Using competitive population evaluation in a differential evolution algorithm for dynamic environments," *European Journal of Operational Research*, vol. 218, no. 1, pp. 7–20, 2012.

[57] P. Novoa-Hernández, D. A. Pelta, and C. C. Corona, *Improvement Strategies for Multi-swarm PSO in Dynamic Environments*. Springer Berlin Heidelberg, 2010, pp. 371–383.

[58] H. Wang, S. Yang, W. H. Ip, and D. Wang, "A particle swarm optimization based memetic algorithm for dynamic optimization problems," *Natural Computing*, vol. 9, no. 3, pp. 703–725, 2010.

[59] V. Wu, Y. Wang, X. Liu, and J. Ye, "Multi-population and diffusion umda for dynamic multimodal problems," *Journal of Systems Engineering and Electronics*, vol. 21, no. 5, pp. 777–783, 2010.

[60] M. Daneshyari and G. G. Yen, "Dynamic optimization using cultural based pso," in *IEEE Congr. Evol. Comput.* IEEE, 2011, pp. 509–516.

[61] V. Noroozi, A. B. Hashemi, and M. R. Meybodi, "Cellularde: A cellular based differential evolution for dynamic optimization problems," in *Adaptive and Natural Computing Algorithms*, A. Dobnikar et al., Ed. Springer Berlin Heidelberg, 2011, pp. 340–349.

[62] P. Novoa-Hernández, C. C. Corona, and D. A. Pelta, "Efficient multi-swarm pso algorithms for dynamic environments," *Memetic Computing*, vol. 3, no. 3, pp. 163–174, Aug 2011.

[63] M. C. du Plessis and A. P. Engelbrecht, "Self-adaptive competitive differential evolution for dynamic environments," in *Symposium on Differential Evolution*. IEEE, 2011, pp. 1–8.

[64] I. Rezazadeh, M. R. Meybodi, and A. Naebi, "Adaptive particle swarm optimization algorithm in dynamic environments," in *Computational Intelligence, Modelling and Simulation*. IEEE, 2011, pp. 74–79.

[65] J. Abbott and A. P. Engelbrecht, "Performance of bacterial foraging optimization in dynamic environments," in *Swarm Intelligence*, M. Dorigo et al., Ed. Springer Berlin Heidelberg, 2012, pp. 284–291.

[66] J. G. O. L. Duhain, "Particle swarm optimisation in dynamically changing environments - an empirical study," Master's thesis, University of Pretoria, Pretoria, South Africa, 2012.

[67] B. Nasiri and M. R. Meybodi, "Speciation based firefly algorithm for optimization in dynamic environments," *Int. J. Artif. Intell.*, vol. 8, no. S12, pp. 118–132, 2012.

[68] A. Sepas-Moghaddam, A. Arabshahi, D. Yazdani, and M. M. Dehshibi, "A novel hybrid algorithm for optimization in multimodal dynamic environments," in *Int. Conf. Hybrid Intell. Syst.* IEEE, 2012, pp. 143–148.

[69] S. Nabizadeh, A. Rezvanian, and M. R. Meybodi, "A multi-swarm cellular pso based on clonal selection algorithm in dynamic environments," in *International Conference on Informatics, Electronics Vision*. IEEE, 2012, pp. 482–486.

[70] S. Nabizadeh, A. Rezvanian, and M. R. Meybodi, "Tracking extrema in dynamic environment using multi-swarm cellular pso with local search," *International Journal of Electronics and Informatics*, vol. 1, no. 1, pp. 29–37, 2012.

[71] A. Sharifi, V. Noroozi, M. Bashiri, A. B. Hashemi, and M. R. Meybodi, "Two phased cellular pso: A new collaborative cellular algorithm for optimization in dynamic environments," in *IEEE Congr. Evol. Comput.* IEEE, 2012, pp. 1–8.

[72] H. Wang, S. Yang, W. Ip, and D. Wang, "A memetic particle swarm optimisation algorithm for dynamic multi-modal optimisation problems," *International Journal of Systems Science*, vol. 43, no. 7, pp. 1268–1283, 2012.

[73] D. Yazdani, M. R. Akbarzadeh-Totonchi, B. Nasiri, and M. R. Meybodi, "A new artificial fish swarm algorithm for dynamic optimization problems," in *IEEE Congr. Evol. Comput.* IEEE, 2012, pp. 1–8.

[74] L. Xiao and X. Zuo, "Multi-depso: A de and pso based hybrid algorithm in dynamic environments," in *IEEE Congr. Evol. Comput.* IEEE, 2012, pp. 1–7.

[75] S. Kundu, D. Basu, and S. S. Chaudhuri, "Multipopulation-based differential evolution with speciation-based response to dynamic environments," in *Swarm, Evolutionary, and Memetic Computing*, B. K. Panigrahi et al., Ed. Springer International Publishing, 2013, pp. 222–235.

[76] T. T. Nguyen, I. Jenkinson, and Z. Yang, "Solving dynamic optimisation problems by combining evolutionary algorithms with kd-tree," in *Conference on Soft Computing and Pattern Recognition*. IEEE, 2013, pp. 247–252.

[77] N. J. Unger, B. M. Ombuki-Berman, and A. P. Engelbrecht, "Cooperative particle swarm optimization in dynamic environments," in *Symposium on Swarm Intelligence*. IEEE, 2013, pp. 172–179.

[78] D. Yazdani, B. Nasiri, A. Sepas-Moghaddam, and M. R. Meybodi, "A novel multi-swarm algorithm for optimization in dynamic environments based on particle swarm optimization," *Appl. Soft Comput.*, vol. 13, no. 4, pp. 2144–2158, 2013.

[79] M. S. Daas and M. Batouche, "Multi-bacterial foraging optimization for dynamic environments," in *Conference of Soft Computing and Pattern Recognition*. Springer Berlin Heidelberg, 2014, pp. 237–242.

[80] J. K. Kordestani, A. Rezvanian, and M. R. Meybodi, "Cdepso: a bi-population hybrid approach for dynamic optimization problems," *Applied Intelligence*, vol. 40, no. 4, pp. 682–694, 2014.

[81] R. Mukherjee, G. R. Patra, R. Kundu, and S. Das, "Cluster-based differential evolution with crowding archive for niching in dynamic environments," *Inf. Sci.*, vol. 267, pp. 58 – 82, 2014.

[82] A. M. Turky and S. Abdullah, "A multi-population electromagnetic algorithm for dynamic optimisation problems," *Appl. Soft Comput.*, vol. 22, pp. 474 – 482, 2014.

[83] ——, "A multi-population harmony search algorithm with external archive for dynamic optimization problems," *Inf. Sci.*, vol. 272, pp. 84 – 95, 2014.

[84] D. Yazdani, B. Nasiri, A. Sepas-Moghaddam, M. Meybodi, and M. Akbarzadeh-Totonchi, "mNAFSA: a novel approach for optimization in dynamic environments with global changes," *Swarm Evol. Comput.*, vol. 18, pp. 38 – 53, 2014.

[85] F. B. Ozsoydan and A. Baykasoglu, "A multi-population firefly algorithm for dynamic optimization problems," in *Conference on Evolving and Adaptive Intelligent Systems*. IEEE, 2015, pp. 1–7.

[86] L. Shen, L. Xu, R. Wei, and L. Cao, "Multi-swarm optimization with chaotic mapping for dynamic optimization problems," in *2015 8th International Symposium on Computational Intelligence and Design (ISCID)*, vol. 2. IEEE, 2015, pp. 132–137.

[87] A. Sharifi, J. K. Kordestani, M. Mahdaviani, and M. R. Meybodi, "A novel hybrid adaptive collaborative approach based on particle swarm optimization and local search for dynamic optimization problems," *Appl. Soft Comput.*, vol. 32, pp. 432 – 448, 2015.

[88] J. K. Kordestani, A. Rezvanian, and M. R. Meybodi, "An efficient oscillating inertia weight of particle swarm optimisation for tracking optima in dynamic environments," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 28, no. 1-2, pp. 137–149, 2015.

[89] N. Fouladgar and S. Lotfi, "A novel approach for optimization in dynamic environments based on modified cuckoo search algorithm," *Soft Comput.*, vol. 20, no. 7, pp. 2889–2903, 2015.

[90] M. Mavrovouniotis, F. Neri, and S. Yang, "An adaptive local search algorithm for real-valued dynamic optimization," in *IEEE Congr. Evol. Comput.* IEEE, 2015, pp. 1388–1395.

[91] D. Yazdani, A. Sepas-Moghaddam, A. Dehban, and N. Horta, "A novel approach for optimization in dynamic environments based on modified artificial fish swarm algorithm," *Int. J. Comput. Intell. Appl.*, vol. 15, no. 02, pp. 1 650 010 – 1 650 034, 2016.

[92] B. Nasiri and M. R. Meybodi, "Improved speciation-based firefly algorithm in dynamic and uncertain environments." *Journal of Information Science and Engineering*, vol. 32, no. 3, pp. 661–676, 2016.

[93] A. Nooraliei, M. R. Meybodi, and B. Masoumi, "Memetic algorithm based on genetic algorithm and improved cuckoo search algorithm for dynamic environment," in *Artificial Intelligence and Robotics*. IEEE, 2016, pp. 54–60.

[94] K. R. Harrison, B. M. Ombuki-Berman, and A. P. Engelbrecht, "A radius-free quantum particle swarm optimization technique for dynamic optimization problems," in *IEEE Congr. Evol. Comput.* IEEE, 2016, pp. 578–585.

[95] J. G. O. L. Duhain and A. P. Engelbrecht, "Towards a more complete classification system for dynamically changing environments," in *IEEE Congr. Evol. Comput.* IEEE, 2012, pp. 1–8.

[96] D. Jia, S. Qu, and L. Li, "A multi-swarm artificial bee colony algorithm for dynamic optimization problems," in *Information System and Artificial Intelligence*. IEEE, 2016, pp. 441–445.

[97] W. Luo, J. Sun, C. Bu, and H. Liang, "Species-based particle swarm optimizer enhanced by memory for dynamic optimization," *Appl. Soft Comput.*, vol. 47, pp. 130 – 140, 2016.

[98] H. Pekdemir and H. R. Topcuoglu, "Enhancing fireworks algorithms for dynamic optimization problems," in *IEEE Congr. Evol. Comput.* IEEE, 2016, pp. 4045–4052.

[99] M. Shakeri and M. Dadvar, "A modified-abc: An explicit-memory based approach with a new memory updating and retrieval in dynamic environments," in *Mexican International Conference on Artificial Intelligence*. IEEE, 2016, pp. 92–98.

[100] Y. Shen, J. Chen, C. Zeng, and L. Wei, "Heterogeneous bare-bones particle swarm optimization for dynamic environments," in *Advances in Swarm Intelligence*, Y. Tan et al., Ed. Springer International Publishing, 2016.

[101] X. Yu and X. Wu, "A multi-point local search algorithm for continuous dynamic optimization," in *IEEE Congr. Evol. Comput.* IEEE, 2016, pp. 2736–2743.

[102] S. K. Nseef, S. Abdullah, A. Turky, and G. Kendall, "An adaptive multi-population artificial bee colony algorithm for dynamic optimisation problems," *Knowledge-Based Systems*, vol. 104, pp. 14 – 23, 2016.

[103] J. K. Kordestani, H. A. Firouzjaee, and M. Reza Meybodi, "An adaptive bi-flight cuckoo search with variable nests for continuous dynamic

optimization problems," *Applied Intelligence*, vol. 48, no. 1, pp. 97–117, 2017.

[104] S. Abdullah, S. K. Nseef, and A. Turky, "An interleaved artificial bee colony algorithm for dynamic optimisation problems," *Connection Science*, vol. 30, no. 3, pp. 272–284, 2017.

[105] A. Boulesnane and S. Meshoul, "Wd2o: a novel wind driven dynamic optimization approach with effective change detection," *Applied Intelligence*, vol. 47, no. 2, pp. 488–504, 2017.

[106] L. Cao, L. Xu, and E. D. Goodman, "A neighbor-based learning particle swarm optimizer with short-term and long-term memory for dynamic optimization problems," *Inf. Sci.*, vol. 453, pp. 463 – 485, 2018.

[107] W. Luo, J. Sun, C. Bu, and R. Yi, "Identifying species for particle swarm optimization under dynamic environments," in *Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2018, pp. 1921–1928.

[108] X. Liu, Z. Zhan, and J. Zhang, "Neural network for change direction prediction in dynamic optimization," *IEEE Access*, vol. 6, pp. 72 649–72 662, 2018.

[109] M. Moradi, S. Nejatian, H. Parvin, and V. Rezaie, "Cmcabc: Clustering and memory-based chaotic artificial bee colony dynamic optimization algorithm," *International Journal of Information Technology & Decision Making*, vol. 17, no. 04, pp. 1007–1046, 2018.

[110] H. Parvin, S. Nejatian, and M. Mohamadpour, "Explicit memory based abc with a clustering strategy for updating and retrieval of memory in dynamic environments," *Applied Intelligence*, vol. 48, no. 11, pp. 4317–4337, 2018.

[111] A. Turky, S. Abdullah, and A. Dawod, "A dual-population multi operators harmony search algorithm for dynamic optimization problems," *Computers & Industrial Engineering*, vol. 117, pp. 19 – 28, 2018.

[112] W. Zhang, M. Zhang, W. Zhang, Y. Meng, and H. Wu, "Innate-adaptive response and memory based artificial immune system for dynamic optimization," *International Journal of Performability Engineering*, vol. 14, no. 9, p. 2048, 2018.

[113] W. Zhang, W. Zhang, G. G. Yen, and H. Jing, "A cluster-based clonal selection algorithm for optimization in dynamic environment," *Swarm Evol. Comput.*, vol. 50, p. 100454, 2019.

[114] Z. Zhu, L. Chen, C. Xia, and C. Yuan, "A history-driven differential evolution algorithm for optimization in dynamic environments," *International Journal on Artificial Intelligence Tools*, vol. 27, no. 06, p. 1850028, 2018.

[115] Z. Zhu, L. Chen, C. Yuan, and C. Xia, "Global replacement-based differential evolution with neighbor-based memory for dynamic optimization," *Applied Intelligence*, vol. 48, no. 10, pp. 3280–3294, 2018.

[116] Y. Yamanaka and T. Tsubone, "Tracking optima in dynamic problems by an optimizer based on piecewise-rotational chaos system," *Nonlinear Theory and Its Applications*, vol. 9, no. 4, pp. 497–516, 2018.

[117] J. K. Kordestani, A. E. Ranginkaman, M. R. Meybodi, and P. Novoa-Hernández, "A novel framework for improving multi-population algorithms for dynamic optimization problems: A scheduling approach," *Swarm Evol. Comput.*, vol. 44, pp. 788 – 805, 2019.

[118] J. K. Kordestani, M. R. Meybodi, and A. M. Rahmani, "A note on the exclusion operator in multi-swarm pso algorithms for dynamic environments," *Connection Science*, pp. 1–25, 2019.

[119] F. Pourpanah, R. Wang, X. Wang, Y. Shi, and D. Yazdani, "mbso: A multi-population brain storm optimization for multimodal dynamic optimization problems," in *Symposium Series on Computational Intelligence*. IEEE, 2019, pp. 672–678.

[120] F. B. Ozsoydan and A. Baykasoğlu, "Quantum firefly swarms for multimodal dynamic optimization problems," *Expert Systems with Applications*, vol. 115, pp. 189 – 199, 2019.

[121] X.-F. Liu, Y.-R. Zhou, X. Yu, and Y. Lin, "Dual-archive-based particle swarm optimization for dynamic optimization," *Appl. Soft Comput.*, p. 105876, 2019.

[122] J. K. Kordestani, A. Rezvanian, and M. R. Meybodi, "New measures for comparing optimization algorithms on dynamic optimization problems," *Natural Computing*, vol. 18, no. 4, pp. 705–720, 2019.

[123] D. Shen, B. Qian, and M. Wang, "A species conservation-based particle swarm optimization with local search for dynamic optimization problems," *Computational Intelligence and Neuroscience*, vol. 2020, 2020.

[124] R. W. Morrison and K. A. D. Jong, "A test problem generator for non-stationary environments," in *IEEE Congr. Evol. Comput.*, vol. 3. IEEE, 1999, pp. 2047–2053.

[125] X. Li and K. H. Dam, "Comparing particle swarms for tracking extrema in dynamic environments," in *IEEE Congr. Evol. Comput.*, vol. 3. IEEE, 2003, pp. 1772–1779.

[126] S. C. Esquivel and C. A. C. Coello, "Particle swarm optimization in non-stationary environments," in *Advances in Artificial Intelligence*, C. Lemaître et al., Ed. Springer Berlin Heidelberg, 2004, pp. 757–766.

[127] D. Parrott and Xiaodong Li, "A particle swarm model for tracking multiple peaks in a dynamic environment using speciation," in *IEEE Congr. Evol. Comput.*, vol. 1. IEEE, 2004, pp. 98–103.

[128] D. Parrott and X. Li, "Locating and tracking multiple dynamic optima by a particle swarm model using speciation," *IEEE Trans. Evol. Comput.*, vol. 10, no. 4, pp. 440–458, 2006.

[129] C. Hu, X. Wu, Y. Wang, and F. Xie, "Multi-swarm particle swarm optimizer with cauchy mutation for dynamic optimization problems," in *Advances in Computation and Intelligence*, Z. Cai et al., Ed. Springer Berlin Heidelberg, 2009, pp. 443–453.

[130] L. Liu, D. Wang, and J. Tang, "Composite particle optimization with hyper-reflection scheme in dynamic environments," *Appl. Soft Comput.*, vol. 11, no. 8, pp. 4626 – 4639, 2011.

[131] T. M. Blackwell, "Swarms in dynamic environments," in *Genet. Evol. Comput. Conf.*, vol. 2723. Lecture Notes in Computer Science, Springer, 2003, pp. 19–26.

[132] J. J. Grefenstette, "Evolvability in dynamic fitness landscapes: a genetic algorithm approach," in *IEEE Congr. Evol. Comput.*, vol. 3. IEEE, 1999, pp. 2031–2038.

[133] H. Richter, "Detecting change in dynamic fitness landscapes," in *IEEE Congr. Evol. Comput.* IEEE, 2009, pp. 1613–1620.

[134] C. Li and S. Yang, "A general framework of multipopulation methods with clustering in undetectable dynamic environments," *IEEE Trans. Evol. Comput.*, vol. 16, no. 4, pp. 556–577, 2012.

[135] T. Zhu, W. Luo, and L. Yue, "Dynamic optimization facilitated by the memory tree," *Soft Comput.*, vol. 19, no. 3, pp. 547–566, 2014.

[136] B. Nasiri and M. R. Meybodi, "History-driven firefly algorithm for optimisation in dynamic and uncertain environments," *International Journal of Bio-Inspired Computation*, vol. 8, no. 5, pp. 326–339, 2016.

[137] T. Zhu, W. Luo, and L. Yue, "Combining multipopulation evolutionary algorithms with memory for dynamic optimization problems," in *IEEE Congr. Evol. Comput.* IEEE, 2014, pp. 2047–2054.

[138] B. Nasiri, M. Meybodi, and M. Ebadzadeh, "History-driven particle swarm optimization in dynamic and uncertain environments," *Neurocomputing*, vol. 172, pp. 356 – 370, 2016.

[139] W. Luo, X. Lin, T. Zhu, and P. Xu, "A clonal selection algorithm for dynamic multimodal function optimization," *Swarm Evol. Comput.*, vol. 50, p. 100459, 2019.

[140] M. C. du Plessis and A. P. Engelbrecht, "Differential evolution for dynamic environments with unknown numbers of optima," *Journal of Global Optimization*, vol. 55, no. 1, pp. 73–99, 2013.

[141] M. C. du Plessis and A. P. Engelbrecht, *Self-Adaptive Differential Evolution for Dynamic Environments with Fluctuating Numbers of Optima*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 117–145.

[142] C. Li, S. Yang, and M. Yang, "An adaptive multi-swarm optimizer for dynamic optimization problems," *Evol. Comput.*, vol. 22, no. 4, pp. 559–594, 2014.

[143] C. Li, T. T. Nguyen, M. Yang, M. Mavrovouniotis, and S. Yang, "An adaptive multipopulation framework for locating and tracking multiple optima," *IEEE Trans. Evol. Comput.*, vol. 20, no. 4, pp. 590–605, 2016.

[144] W. Luo, B. Yang, C. Bu, and X. Lin, "A hybrid particle swarm optimization for high-dimensional dynamic optimization," in *Simulated Evolution and Learning*, Y. Shi et al., Ed. Cham: Springer International Publishing, 2017, pp. 981–993.

[145] D. Yazdani, M. N. Omidvar, J. Branke, T. T. Nguyen, and X. Yao, "Scaling up dynamic optimization problems: A divide-and-conquer approach," *IEEE Trans. Evol. Comput.*, vol. 24, no. 1, pp. 1–15, 2019.

[146] J. Brest, A. Zamuda, B. Boskovic, M. S. Maucec, and V. Zumer, "Dynamic optimization using self-adaptive differential evolution," in *IEEE Congr. Evol. Comput.* IEEE, 2009, pp. 415–422.

[147] P. Korosec and J. Silc, "The differential ant-stigmergy algorithm applied to dynamic optimization problems," in *IEEE Congr. Evol. Comput.* IEEE, 2009, pp. 407–414.

[148] U. Halder, D. Maity, P. Dasgupta, and S. Das, "Self-adaptive cluster-based differential evolution with an external archive for dynamic optimization problems," in *Swarm, Evolutionary, and Memetic Computing*, B. K. Panigrahi et al., Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 19–26.

[149] I. Rezazadeh, M. R. Meybodi, and A. Naebi, "Particle swarm optimization algorithm in dynamic environments: Adapting inertia weight and clustering particles," in *European Symposium on Computer Modeling and Simulation*. IEEE, 2011, pp. 76–82.

[150] C.-K. Au and H.-F. Leung, "An empirical comparison of cma-es in dynamic environments," in *Parallel Problem Solving from Nature*, C. A. C. Coello et al., Ed. Springer Berlin Heidelberg, 2012, pp. 529–538.

[151] S. Biswas, D. Bose, and S. Kundu, "A clustering particle based artificial bee colony algorithm for dynamic environment," in *Swarm, Evolutionary, and Memetic Computing*, B. K. Panigrahi et al., Ed. Springer Berlin Heidelberg, 2012, pp. 151–159.

[152] D. Bose, S. Biswas, S. Kundu, and S. Das, "A strategy pool adaptive artificial bee colony algorithm for dynamic environment through multi-population approach," in *Swarm, Evolutionary, and Memetic Computing*, B. K. Panigrahi et al., Ed. Springer Berlin Heidelberg, 2012, pp. 611–619.

[153] X. Zuo and L. Xiao, "A de and pso based hybrid algorithm for dynamic optimization problems," *Soft Comput.*, vol. 18, no. 7, pp. 1405–1424, 2013.

[154] J. Brest, P. Korosec, J. Silc, A. Zamuda, B. Boskovic, and M. S. Maucec, "Differential evolution and differential ant-stigmergy on dynamic optimisation problems," *International Journal of Systems Science*, vol. 44, no. 4, pp. 663–679, 2013.

[155] U. Halder, S. Das, and D. Maity, "A cluster-based differential evolution algorithm with external archive for optimization in dynamic environments," *IEEE Trans. Cybern.*, vol. 43, no. 3, pp. 881–897, 2013.

[156] P. Novoa-Hernández, C. C. Corona, and D. A. Pelta, "Self-adaptive, multipopulation differential evolution in dynamic environments," *Soft Comput.*, vol. 17, no. 10, pp. 1861–1881, 2013.

[157] R. Mukherjee, S. Debchoudhury, R. Kundu, S. Das, and P. N. Suganthan, "Adaptive differential evolution with locality based crossover for dynamic optimization," in *IEEE Congr. Evol. Comput.* IEEE, 2013, pp. 63–70.

[158] S. Das, A. Mandal, and R. Mukherjee, "An adaptive differential evolution algorithm for global optimization in dynamic environments," *IEEE Trans. Cybern.*, vol. 44, no. 6, pp. 966–978, 2014.

[159] S. Hui and P. N. Suganthan, "Niching-based self-adaptive ensemble de with mmts for solving dynamic optimization problems," in *IEEE Congr. Evol. Comput.* IEEE, 2014, pp. 1536–1541.

[160] Y. Bravo, G. Luque, and E. Alba, "Global memory schemes for dynamic optimization," *Natural Computing*, vol. 15, no. 2, pp. 319–333, 2015.

[161] R. Mukherjee, S. Debchoudhury, and S. Das, "Modified differential evolution with locality induced genetic operators for dynamic optimization," *European Journal of Operational Research*, vol. 253, no. 2, pp. 337 – 355, 2016.

[162] Z. Wang, Z. Zhan, K. Du, Z. Yu, and J. Zhang, "Orthogonal learning particle swarm optimization with variable relocation for dynamic optimization," in *IEEE Congr. Evol. Comput.* IEEE, 2016, pp. 594–600.

[163] R. Takano, H. Sato, and K. Takadama, "Artificial bee colony algorithm based on adaptive local information sharing: Approach for several dynamic changes," in *Genet. Evol. Comput. Conf.* ACM, 2018, pp. 95–96.

[164] Y. Zhou, Y. Yi-Chuan, L. Tmg-Xmg, and Q. Jm, "A hybrid immune algorithm for solving dynamic optimization problems," in *Chinese Control And Decision Conference*. IEEE, 2018, pp. 5326–5332.

[165] R. Vafashoar and M. R. Meybodi, "A multi-population differential evolution algorithm based on cellular learning automata and evolutionary context information for optimization in dynamic environments," *Appl. Soft Comput.*, p. 106009, 2019.

[166] L. Cao, L. Xu, and E. D. Goodman, "A collaboration-based particle swarm optimizer with history-guided estimation for optimization in dynamic environments," *Expert Systems with Applications*, vol. 120, pp. 1 – 13, 2019.

[167] B. Niu, Q. Liu, and J. Wang, "Bacterial foraging optimization with memory and clone schemes for dynamic environments," in *Advances in Swarm Intelligence*, Y. Tan et al., Ed. Springer International Publishing, 2019, pp. 352–360.

[168] X. Luo, Z. Wang, R. Guan, Z. Zhan, and Y. Gao, "A distributed multiple populations framework for evolutionary algorithm in solving dynamic optimization problems," *IEEE Access*, vol. 7, pp. 44 372–44 390, 2019.

[169] R. Chang, H. Hsu, S. Lin, C. Chang, and J. Ho, "Query-based learning for dynamic particle swarm optimization," *IEEE Access*, vol. 5, pp. 7648–7658, 2017.

[170] W. Wu, D. Xie, and L. Liu, "Heterogeneous differential evolution with memory enhanced brownian and quantum individuals for dynamic optimization problems," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 32, no. 02, p. 1859003, 2018.

[171] R. Liaw and C. Ting, "Incorporating fitness inheritance and k-nearest neighbors for evolutionary dynamic optimization," in *IEEE Congr. Evol. Comput.* IEEE, July 2018, pp. 1–8.

[172] W. Luo, R. Yi, B. Yang, and P. Xu, "Surrogate-assisted evolutionary framework for data-driven dynamic optimization," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 3, no. 2, pp. 137–150, 2019.

[173] D. Yazdani, M. N. Omidvar, R. Cheng, J. Branke, T. T. Nguyen, and X. Yao, "Benchmarking continuous dynamic optimization: Survey and generalized test suite," *IEEE Trans. Cybern.*, pp. 1 – 14, 2020.

[174] C. Li and S. Yang, "A clustering particle swarm optimizer for dynamic optimization," in *IEEE Congr. Evol. Comput.* IEEE, 2009, pp. 439–446.

[175] H. G. Cobb and J. J. Grefenstette, "Genetic algorithms for tracking changing environments," in *International Conference on Genetic Algorithms*. Morgan Kaufmann Publishers Inc., 1993, pp. 523–530.

[176] R. Takano, H. Sato, T. Harada, and K. Takadama, "Toward robustness against environmental change speed by artificial bee colony algorithm based on local information sharing," in *IEEE Congr. Evol. Comput.* IEEE, 2015, pp. 1424–1431.

[177] R. Takano, H. Sato, and K. Takadama, "Artificial bee colony algorithm based on adaptive local information sharing meets multiple dynamic environments," *SICE Journal of Control, Measurement, and System Integration*, vol. 12, no. 1, pp. 1–10, 2019.

[178] J. Branke and H. Schmeck, "Designing evolutionary algorithms for dynamic optimization problems," in *Advances in Evolutionary Computing*, A. Ghosh and S. Tsutsui, Eds. Springer Natural Computing Series, 2003, pp. 239–262.

[179] C. Li, T. T. Nguyen, S. Zeng, M. Yang, and M. Wu, "An open framework for constructing continuous optimization problems," *IEEE Trans. Cybern.*, pp. 1–15, 2018.

[180] D. Yazdani, "Particle swarm optimization for dynamically changing environments with particular focus on scalability and switching cost," Ph.D. dissertation, Liverpool John Moores University, Liverpool, UK, 2018.

[181] M. N. Omidvar, X. Li, and K. Tang, "Designing benchmark problems for large-scale continuous optimization," *Inf. Sci.*, vol. 316, pp. 419–436, 2015.

[182] T. T. Nguyen, S. Yang, and J. Branke, "Evolutionary dynamic optimization: A survey of the state of the art," *Swarm Evol. Comput.*, vol. 6, pp. 1 – 24, 2012.

[183] C. Li and S. Yang, "A generalized approach to construct benchmark problems for dynamic optimization," in *Simulated Evolution and Learning*, X. L. et al., Ed. Springer Lecture Notes in Computer Science, 2013, vol. 5361, pp. 391–400.

[184] K. Weicker and N. Weicker, "Dynamic rotation and partial visibility," in *IEEE Congr. Evol. Comput.*, 2000, p. 1125–1131.

[185] R. Tinos and S. Yang, "A framework for inducing artificial changes in optimization problems," *Inf. Sci.*, vol. 485, pp. 486 – 504, 2019.

[186] D. Yazdani, T. T. Nguyen, and J. Branke, "Robust optimization over time by learning problem space characteristics," *IEEE Trans. Evol. Comput.*, vol. 23, no. 1, pp. 143–155, 2018.

[187] C. Bu, W. Luo, and L. Yue, "Continuous dynamic constrained optimization with ensemble of locating and tracking feasible regions strategies," *IEEE Trans. Evol. Comput.*, vol. 21, no. 1, pp. 14–33, 2016.

[188] G. Pamparà and A. P. Engelbrecht, "A generator for dynamically constrained optimization problems," in *Genet. Evol. Comput. Conf.* Association for Computing Machinery, 2019, p. 1441–1448.

[189] X. Yu, Y. Jin, K. Tang, and X. Yao, "Robust optimization over time—a new perspective on dynamic optimization problems," in *IEEE Congr. Evol. Comput.* IEEE, 2010, pp. 1–6.

[190] T. T. Nguyen, "Continuous dynamic optimisation using evolutionary algorithms," Ph.D. dissertation, University of Birmingham, 2011.

[191] R. W. Morrison, "Performance measurement in dynamic environments," in *GECCO workshop on evolutionary algorithms for dynamic optimization problems*, no. 5-8. ACM, 2003.

[192] K. Weicker, "Performance measures for dynamic environments," in *International Conference on Parallel Problem Solving from Nature*. Springer, 2002, pp. 64–73.

[193] J. Branke, *Evolutionary optimization in dynamic environments*. Springer Science & Business Media, 2012, vol. 3.

[194] C. Li, S. Yang, and M. Yang, "Maintaining diversity by clustering in dynamic environments," in *IEEE Congr. Evol. Comput.* IEEE, 2012, pp. 1–8.

[195] L. J. Fogel, A. J. Owens, and M. J. Walsh, "Artificial intelligence through simulated evolution," 1966.

[196] M. Mavrovouniotis, C. Li, and S. Yang, "A survey of swarm intelligence for dynamic optimization: Algorithms and applications," *Swarm Evol. Comput.*, vol. 33, pp. 1 – 17, 2017.

[197] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95-International Conference on Neural Networks*, vol. 4. IEEE, 1995, pp. 1942–1948.

[198] M. R. Bonyadi and Z. Michalewicz, "Particle swarm optimization for single objective continuous space problems: a review," pp. 1–54, 2017.

[199] A. Rakitianskaia and A. P. Engelbrecht, "Training neural networks with pso in dynamic environments," in *IEEE Congr. Evol. Comput.* IEEE, 2009, pp. 667–673.

[200] A. S. Rakitianskaia and A. P. Engelbrecht, "Training feedforward neural networks with dynamic particle swarm optimisation," *Swarm Intelligence*, vol. 6, no. 3, pp. 233–270, 2012.

[201] W. Jatmiko, K. Sekiyama, and T. Fukuda, "A pso-based mobile robot for odor source localization in dynamic advection-diffusion with obstacles environment: theory, simulation and measurement," *IEEE Computational Intelligence Magazine*, vol. 2, no. 2, pp. 37–51, 2007.

[202] W. Jatmiko, A. Nugraha, R. Effendi, W. Pambuko, R. Mardian, K. Sekiyama, and T. Fukuda, "Localizing multiple odor sources in a dynamic environment based on modified niche particle swarm optimization with flow of wind," *WSEAS Transactions on Systems*, vol. 8, no. 11, pp. 1187–1196, 2009.

[203] B. Panigrahi, V. R. Pandi, and S. Das, "Adaptive particle swarm optimization approach for static and dynamic economic load dispatch," *Energy conversion and management*, vol. 49, no. 6, pp. 1407–1415, 2008.

[204] Y. Wang, J. Zhou, Y. Lu, H. Qin, and Y. Wang, "Chaotic self-adaptive particle swarm optimization algorithm for dynamic economic dispatch problem with valve-point effects," *Expert Systems with Applications*, vol. 38, no. 11, pp. 14 231–14 237, 2011.

[205] A. Klyne and K. Merrick, "Intrinsically motivated particle swarm optimisation applied to task allocation for workplace hazard detection," *Adaptive Behavior*, vol. 24, no. 4, pp. 219–236, 2016.

[206] S. A. Abdulkarim and A. P. Engelbrecht, "Time series forecasting with feedforward neural networks trained using particle swarm optimizers for dynamic environments," *Neural Computing and Applications*, pp. 1–17, 2020.

[207] D. J. Kalita and S. Singh, "Svm hyper-parameters optimization using quantized multi-pso in dynamic environment," *Soft Comput.*, vol. 24, no. 2, pp. 1225–1241, 2020.

[208] X. Liu, S. He, Y. Gu, Z. Xu, Z. Zhang, W. Wang, and P. Liu, "A robust cutting pattern recognition method for shearer based on least square support vector machine equipped with chaos modified particle swarm optimization and online correcting strategy," *ISA transactions*, vol. 99, pp. 199–209, 2020.

[209] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 4–31, 2010.

[210] D. Karaboga and B. Akay, "A comparative study of artificial bee colony algorithm," *Applied mathematics and computation*, vol. 214, no. 1, pp. 108–132, 2009.

[211] F. Neri and N. Khan, "Two local search components that move along the axes for memetic computing frameworks," in *Foundations of Computational Intelligence*. IEEE, 2014, pp. 62–69.

[212] M. Srinivas and L. M. Patnaik, "Genetic algorithms: A survey," *computer*, vol. 27, no. 6, pp. 17–26, 1994.

[213] T. Back, F. Hoffmeister, and H.-P. Schwefel, "A survey of evolution strategies," in *Proceedings of the fourth international conference on genetic algorithms*, vol. 2, no. 9. Morgan Kaufmann Publishers San Mateo, CA, 1991.

[214] L. Liu, S. R. Ranjithan, and G. Mahinthakumar, "Contamination source identification in water distribution systems using an adaptive dynamic optimization procedure," *Journal of Water Resources Planning and Management*, vol. 137, no. 2, pp. 183–192, 2011.

[215] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Trans. Evol. Comput.*, vol. 3, no. 2, pp. 82–102, 1999.

[216] T. Back, *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press, 1996.

[217] N. Jin, M. Termansen, K. Hubacek, J. Holden, and M. Kirkby, "Adaptive farming strategies for dynamic economic environment," in *IEEE Congr. Evol. Comput.* IEEE, 2007, pp. 1213–1220.

[218] L. Liu, E. M. Zechman, E. D. Brill, Jr, G. Mahinthakumar, S. Ranjithan, and J. Uber, "Adaptive contamination source identification in water distribution systems using an evolutionary algorithm-based dynamic optimization procedure," in *Water Distribution Systems Analysis Symposium 2006*, 2008, pp. 1–9.

[219] X.-S. Yang *et al.*, "Firefly algorithm," *Nature-inspired metaheuristic algorithms*, vol. 20, pp. 79–90, 2008.

[220] R. M. Thomas, "Survey of bacterial foraging optimization algorithm," *International Journal of Science and Modern Engineering (IJISME)*, vol. 1, no. 4, p. 11, 2013.

[221] K. Socha and M. Dorigo, "Ant colony optimization for continuous domains," *European journal of operational research*, vol. 185, no. 3, pp. 1155–1173, 2008.

[222] D. Yazdani, S. Sadeghi-Ivrigh, D. Yazdani, A. Sepas-Moghaddam, and M. R. Meybodi, "Fish swarm search algorithm: A new algorithm for global optimization," *Int. J. Artif. Intell.*, vol. 13, no. 2, pp. 17–45, 2015.

[223] M. Mahdavi, M. Fesanghary, and E. Damangir, "An improved harmony search algorithm for solving optimization problems," *Applied mathematics and computation*, vol. 188, no. 2, pp. 1567–1579, 2007.

[224] X. Yang and Suash Deb, "Cuckoo search via levy flights," in *Nature Biologically Inspired Computing*. IEEE, 2009, pp. 210–214.

[225] D. Dasgupta, "Advances in artificial immune systems," *IEEE computational intelligence magazine*, vol. 1, no. 4, pp. 40–49, 2006.

[226] P. Mitra and G. K. Venayagamoorthy, "An adaptive control strategy for dstatcom applications in an electric ship power system," *IEEE Transactions on power electronics*, vol. 25, no. 1, pp. 95–104, 2009.

[227] ——, "Real time implementation of an artificial immune system based controller for a dstatcom in an electric ship power system," in *2008 IEEE Industry Applications Society Annual Meeting*. IEEE, 2008, pp. 1–8.

[228] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evol. Comput.*, vol. 9, no. 2, pp. 159–195, 2001.

[229] T. M. Blackwell and P. Bentley, "Improvised music with swarms," in *IEEE Congr. Evol. Comput.*, vol. 2. IEEE, 2002, pp. 1462–1467.

[230] F. van den Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 225–239, 2004.

[231] Z. Zhan, J. Zhang, Y. Li, and Y. Shi, "Orthogonal learning particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 15, no. 6, pp. 832–847, 2011.

[232] E. S. Peer, F. van den Bergh, and A. P. Engelbrecht, "Using neighbourhoods with the guaranteed convergence pso," in *Swarm Intelligence Symposium*. IEEE, 2003, pp. 235–242.

[233] R. Thomsen, "Multimodal optimization using crowding-based differential evolution," in *IEEE Congr. Evol. Comput.*, vol. 2. IEEE, 2004, pp. 1382–1389.

[234] I. L. Schoeman and A. P. Engelbrecht, "A parallel vector-based particle swarm optimizer," in *Adaptive and Natural Computing Algorithms*, B. Ribeiroet al., Ed. Springer Vienna, 2005, pp. 268–271.

[235] Z.-H. Zhan, X.-F. Liu, H. Zhang, Z. Yu, J. Weng, Y. Li, T. Gu, and J. Zhang, "Cloudde: A heterogeneous differential evolution algorithm and its distributed cloud version," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 3, pp. 704–716, 2016.

[236] S. Janson and M. Middendorf, "A hierarchical particle swarm optimizer," in *IEEE Congr. Evol. Comput.*, vol. 2. IEEE, 2003, pp. 770–776.

[237] J. C. Schlimmer and R. H. Granger, "Incremental learning from noisy data," *Machine learning*, vol. 1, no. 3, pp. 317–354, 1986.

[238] A. Tsymbal, "The problem of concept drift: definitions and related work," *Computer Science Department, Trinity College Dublin*, vol. 106, no. 2, p. 58, 2004.

[239] R. Hecht-Nielsen, "Theory of the backpropagation neural network," in *Neural networks for perception*. Elsevier, 1992, pp. 65–93.

[240] R. Brits, A. P. Engelbrecht, and F. van den Bergh, "Locating multiple optima using particle swarm optimization," *Applied Mathematics and Computation*, vol. 189, no. 2, pp. 1859–1883, 2007.

[241] E. M. Zechman and S. R. Ranjithan, "An evolutionary algorithm to generate alternatives (eaga) for engineering optimization problems," *Engineering Optimization*, vol. 36, no. 5, pp. 539–553, 2004.

[242] T. T. Nguyen and X. Yao, "Continuous dynamic constrained optimization—the challenges," *IEEE Trans. Evol. Comput.*, vol. 16, no. 6, pp. 769–786, 2012.

[243] ——, "Dynamic time-linkage problems revisited," in *Workshops on Applications of Evolutionary Computation*. Springer, 2009, pp. 735–744.

[244] R. Z. Farahani, N. Asgari, N. Heidari, M. Hosseininia, and M. Goh, "Covering problems in facility location: A review," *Computers & Industrial Engineering*, vol. 62, no. 1, pp. 368 – 407, 2012.

[245] Q. Gong and R. Batta, "Allocation and reallocation of ambulances to casualty clusters in a disaster relief operation," *IIE Transactions*, vol. 39, no. 1, pp. 27–39, 2007.
[246] H. Zhang, Z. Liang, H. Liu, R. Wang, and Y. Liu, "Ensemble framework by using nature inspired algorithms for the early-stage forest fire rescue—a case study of dynamic optimization problems," *Engineering Applications of Artificial Intelligence*, vol. 90, p. 103517, 2020.
[247] K. M. S. Islam, "Spatial dynamic queueing models for the daily deployment of airtankers for forest fire control," Ph.D. dissertation, University of Toronto, Toronto, Canada, 1998.
[248] J. Brimberg, P. Hansen, N. Mladenovic, and S. Salhi, "A survey of solution methods for the continuous location-allocation problem," *International Journal of Operations Research*, vol. 5, no. 1, pp. 1 – 12, 2008.
[249] S. Senanayake, "Tracking of large crowds with a swarm of aerial robots," Ph.D. dissertation, Monash University, 2015.
[250] C. Huang, Y. Li, and X. Yao, "A survey of automatic parameter tuning methods for metaheuristics," *IEEE Trans. Evol. Comput.*, vol. 24, no. 2, pp. 201–216, 2019.
[251] B. Kiraz, A. Ş. Etaner-Uyar, and E. Özcan, "Selection hyper-heuristics in dynamic environments," *Journal of the Operational Research Society*, vol. 64, no. 12, pp. 1753–1769, 2013.
[252] T. Macias-Escobar, B. Dorronsoro, L. Cruz-Reyes, N. Rangel-Valdez, and C. Gómez-Santillán, "A survey of hyper-heuristics for dynamic optimization problems," in *Intuitionistic and Type-2 Fuzzy Logic Enhancements in Neural and Optimization Algorithms: Theory and Applications*. Springer, 2020, pp. 463–477.
[253] S. A. van der Stockt and A. P. Engelbrecht, "Analysis of selection hyper-heuristics for population-based meta-heuristics in real-valued dynamic optimization," *Swarm Evol. Comput.*, vol. 43, pp. 127–146, 2018.
[254] Y. Tian, R. Cheng, X. Zhang, and Y. Jin, "Platemo: A matlab platform for evolutionary multi-objective optimization [educational forum]," *IEEE Computational Intelligence Magazine*, vol. 12, no. 4, pp. 73–87, 2017.
[255] D. Yazdani, J. Branke, M. N. Omidvar, T. T. Nguyen, and X. Yao, "Changing or keeping solutions in dynamic optimization problems with switching costs," in *Genet. Evol. Comput. Conf.* ACM, 2018, pp. 1095–1102.
[256] Y. Huang, Y. Jin, and K. Hao, "Decision-making and multi-objectivization for cost sensitive robust optimization over time," *Knowledge-Based Systems*, p. 105857, 2020.
[257] Y. Huang, Y. Ding, K. Hao, and Y. Jin, "A multi-objective approach to robust optimization over time considering switching cost," *Inf. Sci.*, vol. 394, pp. 183–197, 2017.
[258] K. Deb, "An efficient constraint handling method for genetic algorithms," *Computer methods in applied mechanics and engineering*, vol. 186, no. 2-4, pp. 311–338, 2000.
[259] C. Raquel and X. Yao, "Dynamic multi-objective optimization: a survey of the state-of-the-art," in *Evolutionary computation for dynamic optimization problems*. Springer, 2013, pp. 85–106.
[260] R. Azzouz, "Evolutionary approaches for dynamic multi-objective optimization," Ph.D. dissertation, Computer Science Department, University of Tunis, 2017.
[261] R. Azzouz, S. Bechikh, and L. B. Said, "Dynamic multi-objective optimization using evolutionary algorithms: a survey," in *Recent advances in evolutionary multi-objective optimization*. Springer, 2017, pp. 31–70.
[262] P. Rohlfshagen, P. K. Lehre, and X. Yao, "Dynamic evolutionary optimisation: An analysis of frequency and magnitude of change,," in *Genet. Evol. Comput. Conf.* ACM, 2009, pp. 1713–1720.

Danial Yazdani (M'20) received his Ph.D. degree in computer science from Liverpool John Moores University, Liverpool, UK, in 2018. He is currently a Research Assistant Professor with the Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China. His main research interests include evolutionary algorithms, dynamic optimization problems, large-scale optimization, and simulation optimization. He was a recipient of the Best Thesis Award from the Faculty of Engineering and Technology, Liverpool John Moores University, and the SUSTech Presidential Outstanding Postdoctoral Award from the Southern University of Science and Technology. He is a member of the IEEE Task Force on Evolutionary Computation in Dynamic and Uncertain Environments, and the IEEE Task Force on Large-Scale Global Optimization.

Ran Cheng (M'16) received the B.Sc. degree from the Northeastern University, Shenyang, China, in 2010, and the Ph.D. degree from the University of Surrey, Guildford, UK, in 2016. He is currently an Associate Professor with the Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China. Dr. Cheng was a recipient of the 2018 and 2021 IEEE Transactions on Evolutionary Computation Outstanding Paper Award, the 2019 IEEE Computational Intelligence Society (CIS) Outstanding Ph.D. Dissertation Award, and the 2020 IEEE Computational Intelligence Magazine Outstanding Paper Award. He is the founding Chair of IEEE Symposium on Model Based Evolutionary Algorithms (IEEE MBEA). He is an Associate Editor of the IEEE Transactions on Artificial Intelligence.

Donya Yazdani received her Ph.D. degree in computer science from the University of Sheffield, Sheffield, UK, in 2020, with a thesis on the time complexity analysis of artificial immune systems for combinatorial optimization. Her current research interests include theoretical analysis of evolutionary algorithms, dynamic optimization problems, and combinatorial optimization. She is currently a lecturer at the Department of Computer Science, Aberystwyth University, Aberystwyth, UK, and a visiting researcher at the University of Sheffield, Sheffield, UK.

Jürgen Branke (M'02) received his Ph.D. degree from the University of Karlsruhe, Karlsruhe, Germany, in 2000. He is a Professor of Operational Research and Systems with the Warwick Business School, University of Warwick, Coventry, UK. He has been an active researcher in the area of evolutionary optimization since 1994 and has published more than 190 papers in international peer-reviewed journals and conferences and a book on "Evolutionary Optimization in Dynamic Environments." Besides dynamically changing environments, his research interests include multi-objective optimization, handling of uncertainty in optimization, simulation-based optimization, and the design of complex systems. He is the editor of ACM Transactions on Evolutionary Learning and Optimization, area editor of the Journal of Heuristics and the Journal on Multi-Criteria Decision Analysis, as well as associate editor of IEEE Transactions on Evolutionary Computation and the Evolutionary Computation Journal.

Yaochu Jin (F'16) is currently a Distinguished Chair, Professor in Computational Intelligence, Department of Computer Science, University of Surrey, Guildford, UK, where he heads the Nature Inspired Computing and Engineering Group. He was a "Finland Distinguished Professor" of University of Jyvaskyla, Finland, a "Changjiang Distinguished Visiting Professor", Northeastern University, China, and "Distinguished Visiting Scholar", University of Technology Sydney, Australia. His main research interests include data-driven evolutionary optimization, evolutionary learning, trustworthy machine learning, and morphogenetic self-organizing systems.

Dr. Jin is presently the Editor-in-Chiefs of the IEEE Transactions on Cognitive and Developmental Systems, and the Complex & Intelligent Systems. He was an IEEE Distinguished Lecturer and Vice President for Technical Activities of the IEEE Computational Intelligence Society. He was the General Co-Chair of the 2016 IEEE Symposium Series on Computational Intelligence and the Chair of the 2020 IEEE Congress on Evolutionary Computation. He is the recipient of the 2018 and 2021 IEEE Transactions on Evolutionary Computation Outstanding Paper Award, the 2015, 2017, and 2020 IEEE Computational Intelligence Magazine Outstanding Paper Award, and the Best Paper Award of the 2010 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology. He was named by the Web of Science as "a Highly Cited Researcher" in 2019 and 2020. He is a Fellow of IEEE.

Xin Yao (M'91–SM'96–F'03) obtained his Ph.D. in 1990 from the University of Science and Technology of China (USTC). He is a Chair Professor of Computer Science at the Southern University of Science and Technology, Shenzhen, China, and a part-time Professor of Computer Science at the University of Birmingham, UK. He is an IEEE Fellow and was a Distinguished Lecturer of the IEEE Computational Intelligence Society (CIS). His major research interests include evolutionary computation, ensemble learning, and their applications to software engineering. His paper on evolving artificial neural networks won the 2001 IEEE Donald G. Fink Prize Paper Award. He also won 2010, 2016, and 2017 IEEE Transactions on Evolutionary Computation Outstanding Paper Awards, 2011 IEEE Transactions on Neural Networks Outstanding Paper Award, and many other best paper awards. He received a prestigious Royal Society Wolfson Research Merit Award in 2012, the IEEE CIS Evolutionary Computation Pioneer Award in 2013 and the 2020 IEEE Frank Rosenblatt Award. He was the President (2014-15) of IEEE CIS and the Editor-in-Chief (2003-08) of IEEE Transactions on Evolutionary Computation.